



A generic cloud migration process model

Mahdi Fahmideh, Farhad Daneshgar, Fethi Rabhi & Ghassan Beydoun

To cite this article: Mahdi Fahmideh, Farhad Daneshgar, Fethi Rabhi & Ghassan Beydoun (2018): A generic cloud migration process model, European Journal of Information Systems

To link to this article: <https://doi.org/10.1080/0960085X.2018.1524417>



Published online: 14 Oct 2018.



Submit your article to this journal [↗](#)



View Crossmark data [↗](#)

A generic cloud migration process model

Mahdi Fahmideh^a, Farhad Daneshgar^b, Fethi Rabhi^c and Ghassan Beydoun^d

^aUniversity of New South Wales, Sydney, Australia; ^bSP Jain School of Global Management, Sydney, Australia; ^cUniversity of New South Wales, Sydney, Australia; ^dUniversity of Technology Sydney, Sydney, Australia

ABSTRACT

The cloud computing literature provides various ways to utilise cloud services, each with a different viewpoint and focus and mostly using heterogeneous technical-centric terms. This hinders efficient and consistent knowledge flow across the community. Little, if any, research has aimed on developing an integrated process model which captures core domain concepts and ties them together to provide an overarching view of migrating legacy systems to cloud platforms that is customisable for a given context. We adopt design science research guidelines in which we use a metamodeling approach to develop a generic process model and then evaluate and refine the model through three case studies and domain expert reviews. This research benefits academics and practitioners alike by underpinning a substrate for constructing, standardising, maintaining, and sharing bespoke cloud migration models that can be applied to given cloud adoption scenarios.

ARTICLE HISTORY

Received 5 December 2016
Revised 25 July 2018
Accepted 4 August 2018

ACCEPTING EDITOR

Par Agerfalk

ASSOCIATE EDITOR

Fredrik Karlsson

KEYWORDS

Cloud computing; cloud migration; legacy systems; metamodel; process model

1. Introduction

Cloud computing technology has received significant attention in addressing the requirements of legacy software systems such as increasing computational power, reducing infrastructure costs, and the efficient utilisation of resources (Armbrust, Fox, & Griffith et al., 2010; Buyya, Chee Shin, & Venugopal, 2008; Koçak, Miranskyy, Alptekin, Bener, & Cialini, 2013). Many organisations are migrating their systems to cloud platforms and many others are moving from one existing cloud platform to another in a post-migration phase. The global cloud computing market continues to grow from \$40.7 billion in 2011 to an expected \$241 billion in 2020 (Ried, Kisker, Matzke, Bartels, & Lisserman, 2011).

Concomitant to this growth has been a volume of research conducted by both academia and industry. The research studies vary from the purely technical to the highly theoretical ones analysing organisational impact of the cloud technology (Oliveira, Thomas, & Espadanal, 2014; Venters & Whitley, 2012; Yang & Tate, 2012). Many new concepts have thus been banded around the cloud computing domain. These concepts relate to various migration activities from planning, platform selection, reengineering, code refactoring, and testing to legacy system deployment on cloud platforms. Depending on the research source, concepts are expressed using different terms, fragmented, or even merged in the literature.

As with any new emerging field, the interwoven tapestry of languages is initially profitable in allowing an interpretative and a creative discourse in the inception phase of the field. But, as the field of cloud computing

somewhat matures, a consensual view of the cumulative research that binds and integrates all the views together is more efficacious for knowledge sharing (Hollenbeck, 2008; Whetten, 1989). Developing that view requires a conceptual foundation which is not yet available. This gap hinders knowledge interoperability and critical information sharing among scholars and/or practitioners involved in cloud migration scenarios (M. Hamdaqa & Tahvildari, 2012; O. Zimmermann, Miksovich, & Küster, 2012). Making a separation between the definition of a cloud migration process and technical platform-specific operationalisation details is also becoming more pressing (Hamdaqa, Livogiannis, & Tahvildari, 2011). Indeed, as noted in Fahmideh, Daneshgar, Low, and Beydoun (2016), the time is ripe for providing a more abstract view of the current chaotic state of cloud migration.

There are many common concepts incorporated into existing models related to migrating systems to the cloud. Although they may not have been expressed in an identical way, it can be helpful if these concepts such as phases and activities are factored out into one, or at least be subsumed by one, unified model at a convenient level of abstraction. Each year, a considerable number of models are suggested, each represents a different viewpoint of the same conceptualisation. This in itself is an indication that the field has reached a maturity point where the development of such a generic reference model becomes timely and important.

Metamodels capture common concepts, relationships, and ways of working. They are often suggested for achieving knowledge interoperability and integration of a domain of interest (Atkinson & Kuhne,

2003; Gonzalez-Perez & Henderson-Sellers, 2008). In essence, they provide a language infrastructure to freely describe the domain in a way that users can better understand it (Rossi & Brinkkemper, 1996). The significance of metamodels as a way for abstracting cloud computing concepts has been already emphasised in the community (Hamdaqa & Tahvildari, 2012; Leymann, 2011; Loutas, Kamateri, Bosi, & Tarabanis, 2011). In line with this view, the objective of this paper is to *develop and evaluate a generic metamodel that captures and harmonises common process elements of cloud migration process and that can be used to create, standardise, and share situation-specific cloud migration models*. Using the literature, we identified and distilled common concepts and integrated them into a process metamodel which is evaluated and refined using industry cloud migration exemplars. The resultant metamodel is cloud computing specific but context agnostic. It can be grounded and extended to be adapted to a given scenario and provides a basis for method engineers to define, configure, and share any migration methodological knowledge for managing cloud migration endeavours.

Demonstration and evaluation section which illustrates the expressiveness of the metamodel in representing enacted cloud migration processes and discusses the expert appraisal of the metamodel. This is followed by the Demonstration section which illustrates the expressiveness of the metamodel in representing enacted cloud migration processes. Next, the evaluation of the metamodel is presented. Finally, the paper concludes with a discussion on the implications and limitations of this study.

2. Background and related work

2.1. Legacy systems and cloud migration

Legacy systems are often characterised by maintenance challenges. A notable early definition of them is given by Bennett (1995) as “large software systems that we don’t know how to cope with but are vital to our organisation” (p. 19). They are costly to maintain, are inflexible to changes, are difficult to be integrated with other systems, and have outdated documentation. Nevertheless, they are major components of IT-based organisations, providing business services, organisational knowledge, and a significant competitive advantage (Bennett, 1995; Erlikh, 2000; Sneed, 1995).

The migration phenomenon is about the physical movement of people, i.e., migrants, from one geographic location to another for a certain period of time (Clark, 1986). Migration can be taken for a short term or long term, short distance or long distance, or voluntary or obligatory and have some permanence, clear source, and target locations (Lee, 1966). Studying cloud migration as an instance of switching from on-

premise hosted systems to cloud-hosted systems has been studied from various perspectives. Some studies centre on the development of models aiding organisations to decide the suitability of cloud adoption (Misra & Mondal, 2011). Others provide tools aiding for making decisions about selecting suitable cloud services (Khajeh-Hosseini, Greenwood, Smith, & Sommerville, 2012). Others highlight inhibitors and enablers of using cloud services (Oliveira et al., 2014) and benefits of cloud adoption in terms of enhanced competitive advantages (Truong & Dustdar, 2010) or service interoperability issues (Toosi, Calheiros, & Buyya, 2014).

Often a cloud migration process involves many concept variants and several ways of instantiation. This process itself is contingent on existing organisational structures and characteristics of a legacy system. It is common wisdom that no two cloud migration scenarios are exactly the same and every scenario requires its own specific management process. Failures in migration scenarios are often due to poor understanding of computing requirements, early engagement with the technical implementation of a cloud solution, and facing unexpected issues that were out of the control of service consumers and providers (Chow, Golle, & Jakobsson et al., 2009; Linthicum, 2012; Pepitone, 2011; Tsidulko, 2016).

A model at a conceptual level which aims at identifying core domain concepts and their relationships can help zoom in and provide a foundation for the representation and maintenance of bespoke cloud migration models. This not only assists method engineers in managing complexity but also allows sharing method knowledge among varying cloud computing communities. More recently, Fahmideh et al. (2016) have reviewed existing migration models and found that each one comes with its own concepts with a varying focus such as reusing legacy system codes (Menychtas, Santzaridou, & Kousiouris et al., 2013), addressing interoperability issues (Mohagheghi, Berre, Henry, Barbier, & Sadovykh, 2010), and finding optimum distributions of system components on cloud servers (Frey & Hasselbring, 2011; Menzel & Ranjan, 2012). Beyond this technical material, which is still important, we are yet to see a research effort that provides an integrated picture of the various methodological concepts. Tying those fragmented works within the literature and making an integrated view of intellectual bins is certainly compelling. Metamodelling is a clearly plausible approach as we shall discuss next.

2.2. Metamodelling

A metamodel is “a model of a model or a model of a collection of models” (Atkinson & Kuhne, 2003). Raising the level of abstraction in modelling systems along with advantages such as improved reusability,

interoperability, and reduced system development time has resulted in the emergence of a large number of metamodels. To provide a synopsis of notable literature on using metamodels to facilitate the use of cloud computing technology, we identified four streams as follows.

The first stream concentrates on abstracting the technical aspects of a cloud computing architecture such as multi-tenancy, elasticity, and data security. Studies such as Hamdaqa et al. (2011), Liu, Tong, and Mao et al. (2011), Zhang and Zhou (2009), and Zimmermann, Pretz, Zimmermann, Firesmith, and Petrov (2013) and white papers published by major cloud computing players such as IBM, HP, Oracle, and Cisco are subsumed under this classification. Capturing the common knowledge of designing solution architectures has been the topic of discussions in Fehling, Leymann, Rüttschlin, and Schumm (2012) and Fehling and Retter (2011) in which researchers propose a catalogue of software patterns for integrating legacy source codes with third-party cloud services.

The second stream uses metamodels as a way for sharing green cloud computing practices such as reducing energy consumption and carbon emission of data centres (Procaccianti, Lago, & Lewis, 2014). Another work proposes a metamodel of the green practice for business processes leveraging cloud services (Nowak, Breitenbücher, & Leymann, 2014). These include classes of patterns for environmental impact, pollution, and waste. Dougherty, White, and Schmidt (2012) also propose a metamodel-based auto-scaling resource management to improve server utilisation and reduce idle time compared with over-provisioned servers.

The third stream is concerned with quality aspects of cloud services. A metamodel developed by the A4Cloud project captures the knowledge related to non-functional properties of cloud services and how they influence the accountability of their providers (Nunez, Fernandez-Gago, Pearson, & Felici, 2013). The purported goal of the metamodel is to act as a language to model cloud service accountability in terms of transparency, verifiability, observability, and liability from which metrics are derived to monitor the quality of cloud services. The proposed metamodel by Cimato, Damiani, Zavatarelli, and Menicocci (2013) models concepts related to the certification process of cloud services. Keller and König (2014) and Martens and Teuteberg (2011) respectively propose metamodels to model risks and compliance efforts for cloud computing as a socio-technical artefact.

The fourth stream uses model-driven techniques to represent legacy system variabilities combined with transformation to a given target cloud platform. For

example, studies by Ardagna, Nitto, and Casale et al. (2012) and Wettinger, Behrendt, and Binz et al. (2013) address the issue of migrating a legacy system across different cloud platforms using metamodel transformation techniques. Research in this direction has also resulted in languages in areas such as risk modelling (Zech, Felderer, & Breu, 2012), service compliance management (Brandic, Dustdar, & Anstett et al., 2010), cryptography (Bain, Mitchell, Sharma, Stefan, & Zimmerman, 2011), distributed data-parallel computing (Isard & Yu, 2009), cloud-mobile hybrid applications (Ranabahu, Maximilien, Sheth, & Thirunarayan, 2011), big data analytic algorithms (Weimer, Condie, & Ramakrishnan, 2011), automatic code generation (Sledziewski, Bordbar, & Anane, 2010), and maximising reusability of SaaS (software as a service) (La & Kim, 2009). The central claim of these technical studies is on the seamless transformation of legacy system codes to different cloud platforms using model transformation techniques. The current research develops a metamodel that raises the abstraction level to cloud migration process.

3. Research method

3.1. Overview

We developed our metamodel using the design science research paradigm (Gregor & Hevner, 2013; Peffers, Tuunanen, Rothenberger, & Chatterjee, 2008). Design science research typically involves developing new artefacts, constructs, models, methods, or instantiations to address organisational IT problems. We conducted phases proposed by Peffers et al. (2008). As shown in Figure 1, the input to each phase was the metamodel resulted from the predecessor phase. We conducted the following phases in the one iteration.

3.1.1. Problem identification and objective definition

The proposed metamodel is addressing an important and timely problem with respect to a constituent community: different and heterogeneous viewpoints of the same process of legacy system migration to cloud platforms. Each viewpoint is expressed with different terms that are narrow in focus. There is currently no established mapping between these viewpoints to attain a harmonised overarching view. The proposed formative metamodel describing the process required for moving legacy systems to cloud platforms can be potentially a candidate for future method tailoring and interoperability in a consistent and systematic manner. It is agnostic to both the target cloud platforms and the domain of legacy systems.

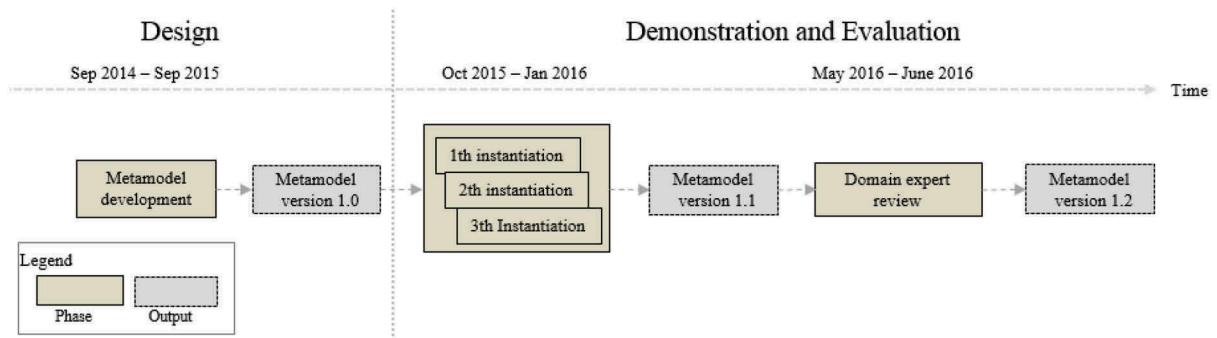


Figure 1. Design science research process specialised for this research.

3.1.2. Design

We identified a set of commonly used process concepts along with their definitions and relationships from the literature on cloud migration. The differences between definitions were reconciled into a consistent and coherent set of concepts. These concepts were grouped based on their similarities/context and then organised into a generic process metamodel. The outcome of this phase was the first version of the metamodel, version 1.0.

3.1.3. Demonstration and evaluation

We have used two methods for the validation. First, the phase examined the expressiveness power of the first metamodel version 1.0 in representing process elements of three projects. The selection of the projects for the analysis was based on (i) having clear goals of cloud migration; (ii) adopting various service delivery models such as IaaS (infrastructure as a service), SaaS (software as a service), and PaaS (platform as a service); and (iii) availability of projects' databases including documentation, diagrams, notes, and codes related to the conducted scenario. The metamodel was found deficient with respect to some new concepts that were earlier overlooked. Support for these new concepts was added to the metamodel. This yielded the next version 1.1 of the metamodel.

Next, the metamodel was reviewed by a group of domain experts. Based on input from our interviewees and other informal communications with different practitioners and academics working in cloud computing, we noted that identifying expertise for the purpose of metamodel evaluation was somewhat difficult. Such expertise is often misconstrued and labelled in the midst of a myriad of other IT areas of expertise. This is attributed to two reasons: (i) the fact that much of legacy system migration to cloud projects are conducted partially due to barriers such as security issues and unwanted organisational changes and (ii) a subjective interpretation of the cloud migration process where some people viewed it as merely virtualisation and others deemed it as huge legacy code refactoring. Considering every willing person to conduct

metamodel evaluation was clearly not scientifically sound. Our general rule-of-thumb was the selection of experts who had hands-on experience and been directly involved throughout cloud migration projects as a programmer, system architect, or project manager for at least one year. Alternatively, an expert could also be an academic with scientific publications in peer-reviewed journals in the cloud computing field related to migrating legacy systems to cloud computing platforms. We carefully scanned profiles of cloud computing experts in the social media such as Twitter, LinkedIn, Facebook, and academic research groups. We recruited only if the identified professional had experience in the roles above and/or high-level academic knowledge on legacy system migration to the cloud. For selected experts, the metamodel description (a 25-page document detailing the metamodel along with a list of open-ended questions about the metamodel) and an invitation letter were sent to the expert in a subsequent communication email. We identified four experts from different countries who were interested in conducting the metamodel review. They all had leading cloud migration roles exceeding 7 years. The experts were not aware of and did not communicate each other. Their feedback was analysed and used to refine the metamodel. During the analysis, clarifications were sought as needed to prevent any misinterpretation of their comments. The output of this phase resulted in the final version of later metamodel presented in Figure 2.

3.2. Design phase

3.2.1. Developing design principles for the metamodel

Assuring the quality of a metamodel is an integrated part of the actual metamodeling process which bears on the metamodel ability to satisfy stated needs. Lindland, Sindre, and Solvberg (1994) proposed general design principles (DPs) underlining a metamodeling endeavour. These are the following: syntactic adequacy, tailorability, and comprehensibility. The way each principle has been applied in the context of this research is detailed in what follows.

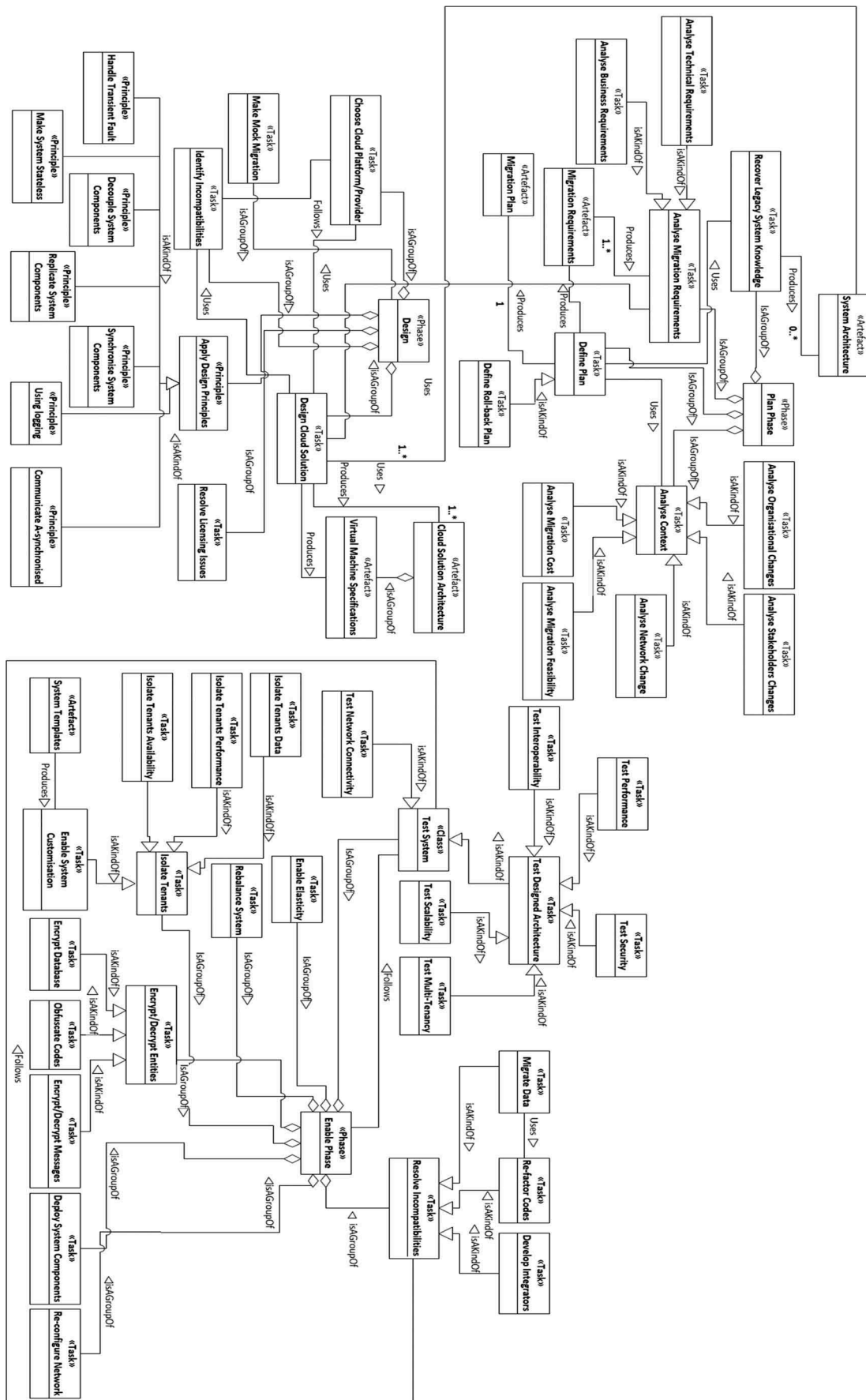


Figure 2. The proposed metamodel.

Semantic adequacy (DP1) is the correspondence between concepts in the metamodel and the domain of interest, i.e., cloud migration processes. To ensure

this, two evaluation criteria are applied during the metamodeling process: *completeness* and *validity*. Completeness is the extent to which a metamodel

can be used to make statements about the domain. Validity is the correctness of statements and their relevance to the domain. Achieving full adherence to this DP may not be practical, but an appropriate coverage of core process concepts incorporated during a typical transition is important. In this research, a good yardstick to get a feasible semantic quality is key functional and non-functional methodological requirements specific to cloud migration as elaborated in Fahmideh et al. (2016). These include, for example, analysing organisational context, identifying computational requirements, understanding legacy system architecture, and the choice of target cloud platform. Additionally, a metamodel needs specifying relationships among process components such as sequences, associations, and aggregations. For example, according to Fahmideh et al. (2016), a common challenge in migrating systems to the cloud is incompatibilities (e.g., data or functions) between legacy systems and cloud services. That is, for a chosen cloud platform, a sequence of steps in the migration process is required to identify any incompatibilities between platforms. Relationships defined in our proposed metamodel are based on recommendations in the literature. Our overall definition of semantic adequacy is as follows: *the proposed metamodel should capture all important relevant concepts for the incorporation into a typical transition process of legacy systems to cloud platforms.*

Tailorability (DP2) is the extent to which a metamodel can be customised and extended to address new requirements. A process metamodel allows tailoring into different models. Tailorability is required as the integrating legacy systems with cloud services may be undergone by several factors such as the choice of a target cloud platform, reusability of legacy system codes, security requirements, and system workload. These factors and many others influence tailoring decisions. To support tailorability, the metamodel should have some modularity in a way that different subsets of concepts can be selected and put together to fit needs of a particular scenario (Cameron, 2002). Another prerequisite for the tailorability is the fact that the more a metamodel is close to the problem domain, the simpler is its customisation and specialisation (Jonkers, Stroucken, & Vdovjak, 2006). Therefore, another key attribute of an effective process metamodel is as follows: *the metamodel should be tailorable to different cloud migration scenarios.*

Pragmatic quality (DP3) is the extent to which a metamodel is comprehended by its audience (Lindland et al., 1994). A metamodel is expected to have understandable concepts, to reflect intentions of its audience, to minimise multiple interpretations, and to avoid unnecessary modelling details (Ambler, 2005). This quality is largely determined by the

quality of its diagrams, icons, names, and definitions (Lindland et al., 1994). In the context of this research, the third DP is formulated as follows: *the definitions, names, and relationships of concepts in the metamodel should be comprehensible by cloud computing domain experts.*

Metamodels have the potential to become too complex if they include a large number of concepts, definitions, and relationships. In deciding on the level of metamodel complexity, every designer faces a trade-off between tailorability, understandability, and comprehensiveness (Gonzalez-Perez and Henderson-Sellers, 2008). If the designer tends to maximise the metamodel tailorability, then abstracting out and making domain concepts variable instead of being fixed take precedence over understandability. On the other hand, adherence to the understandability principle pushes towards making the metamodel more detailed and elaborated at the expense of the tailorability. Making a trade-off among these principles, e.g., too generic or too specific metamodel, is always a difficult issue to decide (Henderson-Sellers & Gonzalez-Perez, 2010). For example, bearing in mind the priority of the metamodel understandability over the completeness, the designer may not include many different complex domain concepts and relationships in the metamodel to make a more generic and less-detailed metamodel. In the current study, results identified from examining the metamodel adherence to its purported DPs through conducting the case study analysis and domain expert review (Sections 5 and 6) show that the metamodel is not a complicated entity to use.

3.2.2. Metamodel development

A brief explanation of steps undertaken to develop our initial metamodel is explained in what follows, though a more detailed description is available at Fahmideh, Daneshgar, Beydoun, and Rabhi (2017b). To create the metamodel, we conducted a systematic literature review (Kitchenham, Pearl Brereton, & Budgen et al., 2009) as a point of departure for identifying cloud migration process concepts. These concepts could be a (i) *task*: a discrete and small unit of work performed by developers to achieve specified goals, (ii) *work-product*: a significant artefact as a result of performing tasks, (iii) *principle*: a consideration that should be taken into account during design of a cloud solution architecture, and (iv) *phase*: a collection of concepts logically classified to provide a high-level organisation to the cloud migration process.

Adherence to the DPs has been interleaved with the actual metamodeling process to ensure that the metamodel satisfies stated needs. To address DP1, we had a tendency in selecting concepts that were cloud-platform independent and sufficiently generic to a variety of cloud migration scenarios. Concepts that

were too general or belonged to the traditional software development process were not incorporated as they were deemed out of the scope of this research. The full list of all the identified studies along with the extracted concepts is presented in [Appendix A](#). For DP2, it was critical in the metamodeling process that concepts are chosen at the right abstraction layer. Through a bottom-up approach, all concepts were grouped based on their similarities and definitions to derive a new set of high-level overarching concepts. Various definitions of concepts were reconciled to reach a set of internally consistent metamodel concepts. When there were several definitions for a concept, a hybrid definition encompassing all definitions was synthesised. The relationships among all process concepts such as sequence, association, specialisation, and aggregation were revisited as needed. For DP3, the choice of concepts' names, definitions, and terms was made in a way to be intuitively understandable for users. A simple version of Unified Modelling Language (UML) notation UML (2004), which is common for information modelling, was used to represent the metamodel in a well-structured manner.

4. Resultant metamodel

In the following, we provide an overview of the metamodel though a more detailed technical description of its internal working can be found at Fahmideh, Daneshgar, Beydoun, and Rabhi (2017a). The metamodel includes a set of concepts that are commonly performed in the cloud migration. They are organised into three phases namely *Plan*, *Design*, and *Enable*. Operationalisation details of concepts are left to each individual instantiation of the metamodel using available implementation techniques in the cloud computing literature or tools in marketplace. [Figure 2](#) shows the metamodel along with definitions of the concepts presented in [Table 1](#).

The *plan* phase starts with a feasibility analysis of adopting cloud services in terms of potential changes in organisational structure, local network, and cost saving. The legacy system architecture and its functional and non-functional requirements are identified. This can be a deployment model of the system in the local network of organisation. This model helps in estimating required effort to make the system cloud-enabled. Legacy systems may have certain requirements that can be satisfied by utilising cloud services such as computational, storage space, or security requirements. The phase also includes preparing a plan which organises the sequence of activities in the course of a migration process.

In *design* phase, a new architectural model showing how legacy system components utilise cloud services is produced. The re-architecting process includes

identifying suitable components for moving to and re-deployment in the cloud in order to satisfy non-functional requirements such as data security, performance variability, acceptable network delay, and scaling latency. In re-architecting legacy systems to the cloud, DPs play a central role. For instance, system components should minimise dependency and store the contextual data during their execution in order to support the individual scalability feature. An important consideration during cloud architecture design is the performance variability of cloud servers and latency between a local network and cloud servers. Developers should implement mechanisms in legacy systems to detect and handle transient faults that may occur in the cloud. A key work-product of this phase is a new skeletal architecture specifying an optimum distribution of legacy components on the cloud servers with respect to non-functional requirements.

The *enable* phase is to implement the architectural model designed in the previous phase. Legacy systems might have been implemented with technologies that are not compatible with cloud services. If this situation occurs, such incompatibilities should be identified and accordingly resolved through adaptation mechanisms such as refactoring source codes of legacy systems, modifying data, and wrappers. Legacy systems might not have been implemented with the support for dynamic resource acquisition and release under input workload. Instead, new physical servers are added to address workload. Mechanisms for system elasticity are implemented by continuous system monitoring and performing actions for resource management based on scaling rules triggered in a specific workload threshold, event, or metric. The phase may entail either adding new components to the new legacy system architecture or having them separately hosted on cloud servers. Additionally, the local network is reconfigured to provide access to cloud services. If required, legacy components and third-party tools are installed. Finally, both functional and non-functional aspects of the migrated system are tested.

5. Demonstration and evaluation: retrospective case studies

As the proposed metamodel is sufficiently generic and covers key domain concepts, it is anticipated that real-world migration processes to be representable using the metamodel concepts. To appraise the metamodel adherence to DP1 and DP2, we used three case studies presented in [Table 2](#). We traced the conformance of the concepts in these scenarios to the corresponding ones in the metamodel. This was performed by grouping and mapping concepts in these scenarios to the metamodel concepts according to their relevance. Some of the leading questions (see [Appendix B](#)) that were used during the case review were as follows: (i) what activities were performed and deliverables produced by the developers during

Table 1. Key process concepts incorporated into a typical process of legacy system migration to the cloud.

Concept	Definition
Analyse migration requirements	Identify a set of requirements to be satisfied by the cloud such as computational requirements, data storage, security, response time, and elasticity.
Define plan	Define a sequence of tasks that guide the migration process by analysing feedback from stakeholders. A plan may include (i) notice of temporal unavailability of legacy systems, (ii) roll-back the system to in-house versions, (iii) migration type such as complete or partial, and (iv) legacy system retirement procedures.
Recover legacy system knowledge	Produce a complete representation of legacy system architecture including its data, components, dependencies among components and infrastructure, system data usage, and resource utilisation model (e.g., CPU, Network, and storage).
Choose cloud platform/provider	Define a set of suitability criteria that characterise desirable features of cloud providers including pricing model, constraints, offered QoS, electricity costs, power and cooling costs, organisation migration characteristics (migration goals and available budget), and system requirements.
Design cloud solution	Identify legacy system components with respect to migration requirements and then define their distribution cloud servers.
Identify incompatibilities	Identify incompatibilities between legacy system components and cloud services.
Make system stateless	Enable the legacy system to handle safety and traceability of tenant's session when various system instances hosted in the cloud.
Decouple system components	Decouple system components from each other. Use mediator and synchronisation mechanisms to manage interaction between the loosely coupled components.
Replicate system components	Partition and deploy legacy system components (e.g., database and business logic) on multiple cloud servers.
Make mock migration	Build a prototype of new cloud solution to get an understanding of how the functional and none-functional aspects of the system will work in the cloud.
Use logging	Use logging mechanism to facilitate system debug and resource monitoring when running in the cloud.
Resolve licensing issues	Define and monitor a pay-as-you-go licensing model to handle unintended licence agreement violations due to automatic scaling.
Develop integrators	Develop mediators/wrappers to hide incompatibilities occurring at runtime between legacy system components and selected cloud services that are plugged to these system components.
Deploy system component	Instal system components and any required third-party tools in the cloud.
Enable elasticity	Define scaling rules and provide support for dynamic acquisition and release of cloud resources.
Encrypt database	Encrypt critical databases prior to hosting in the cloud.
Handle transient faults	Detect and handle transient faults may occur in the cloud.
Isolate tenant	Protect tenants' data, performance, and faults from other tenants, which are running on the same cloud server.
Encrypt/decrypt messages	Secure message transmission between the local components and those hosted in the cloud or distributed across multiple clouds using an encryption mechanism.
Obfuscate codes	Protect unauthorised access to code blocks of components by other tenants that are running on the same cloud provider.
Reconfigure network	Reconfigure the running environment of the system including reachability policies to resources and network, connection to storages, setting ports and firewalls, and load balancer.
Synchronise/replicate system components	Provide support in the system to synchronise multiple components (e.g., database replica) hosted on premise network and cloud servers.
Communicate a-synchronous	Enable application components to interact in an asynchronous manner.
Test system	Test system security, interoperability, multi-tenancy, performance, scalability, and network connectivity of the system that migrated to the cloud.

Table 2. Description of case studies

Case 1: <i>InformalT</i> (Sweden)	Case 2: <i>TOAS</i> (Finland)	Case 3: <i>SpringTrader</i> (US)
<i>InformalT</i> is a software development company providing digital document processing services. The document comparison (DC) system, developed by the company, is a Web-based enterprise solution for enhancing document management processes. DC provided a fast and easy way to compare textual and graphical content of different digital documents. DC was originally designed to offer services to medium and large organisations which had adequate infrastructure and staff to instal and run the system. <i>InformalT</i> wondered expanding DC's services around small publishing companies. However, small companies could not afford DC as they would need a high financial commitment for installation and paying and usage cost of users. Cloud enablement of the system could facilitate an efficient and agile maintenance environment for the DC for small companies.	TietoOyj is a software development company that has built an open-source platform called Tieto Open Application Suite (TOAS) that provides an integrated set of middle-wares, tools, and services for developing new software systems and deploying on the cloud. The platform aims increasing development speed, automation, and the integrity of cloud-based software systems. A cloud migration project was launched by Tieto to migrate a legacy batch processing system to this TOAS platform. The outdated hardware infrastructure and software platform of the legacy system was the key driver to move to TOAS which leads enhanced system performance and reduced infrastructure cost.	SpringTrader is an open-source Web-based system that has been originally developed by Pivotal company and maintained by many contributors over time. The system allows registered users to monitor and manage a portfolio of stocks, lookup stock quotes, and buy/sell stock shares. Pivotal company recently has developed its private cloud platform, which named Pivotal Cloud Foundry. The platform is an open-source platform for developing and deploying portable cloud-native enterprise systems. Pivotal decided to move SpringTrader system to this new cloud platform to enable users to access real-time stock market data in more interactive way, to individually scale up/down each system component (also called micro services), and to improve system maintainability.

each phase of the migration project? (ii) what cloud-specific challenges were that those developers faced in each phase?

Inspired by the previous studies suggesting the worthiness of secondary data in the assessment of metamodels (Antkiewicz, Czarnecki, & Stephan, 2009; Beydoun, Low,

Table 3. Support of concepts and relationships in the migration scenarios by the metamodel (√:instantiated *:not instantiated)

	Name	InformaIT	TOAS	SpringTrader
Metamodel concepts	Recover legacy system knowledge	√	√	√
	Choose cloud platform/provider	√		
	Design cloud solution	√	√	√
	Identify incompatibilities	√	√	√
	Decouple system components	√	√	√
	Adapt data	√		√
	Develop integrators		√	√
	Refactor codes			√
	Reconfigure network		√	
	Relationship	InformaIT	TOAS	SpringTrader
Metamodel relationships	Design cloud solution <i>Uses</i> Analyse migration requirements		√	
	Design cloud solution <i>Uses</i> Identify incompatibilities	√	√	√
	Design cloud solution <i>Uses</i> Choose cloud platform/provider	√	√	√
	Refactor codes <i>Uses</i> Identify incompatibilities	√	-	√
	Design cloud solution <i>Uses</i> Recover legacy system knowledge			√
	Refactor codes <i>Uses</i> design cloud solution	√		√
	Migrate database <i>Uses</i> Refactor codes	√		
	Test system <i>Uses</i> Design cloud solution	√	√	√
	Plan migration <i>Follows</i> Design phase	√	√	√
	Design phase <i>Follows</i> Enable phase	√	√	√
	Choose cloud provider <i>Follows</i> Identify incompatibilities	√	√	√

& Henderson-Sellers et al., 2009; Othman & Beydoun, 2013), we used project documents from a variety of sources (e.g., system models, codes, and user histories) to obtain a better understanding of the enacted in-house method. The summarised results of the tracing in Table 3 show the extent to which metamodel adheres to DP1 and DP2. In this table, the first column shows a metamodel concept, and the next three columns show corresponding instantiation of the concept in three case study scenarios.

5.1. Within-case analysis: InformaIT case

The following paragraphs describe how the concepts in the metamodel are instantiated and specialised to represent tasks that were carried out by the development team in InformaIT project (Rabetski, 2012). The unit of analysis is the legacy system. The 43-page secondary document of this project was carefully reviewed. Figure 3 depicts instances of enacted metamodel concepts in InformaIT highlighted with grey colour.

As one of the first tasks, the developers performed *preliminary analysis* to identify benefits and risk of migrating the system to the cloud in terms of privacy, vendor lock-in, and environmental limitations. This activity is an instantiation of the concept *analyse migration feasibility* in the metamodel. Additionally, an activity, called *current DC implementation*, was performed to identify the current deployment model of DC. The metamodel supports this task through an instance of *recover legacy system knowledge* defined in the plan phase.

The developers estimated the cost of the migration project on the basis of required server instances, storage, data transfer, storage transaction, cache, and database. They realised that the cost could be down from \$764.99

in the cloud model compared to \$1264.99 in the legacy system model when leveraging elastic scalability. The abovementioned cost analysis in InformaIT can be derived from the *analyse migration cost* in the metamodel which is a subclass of *analyse context*.

Once the cloud migration was perceived as a viable solution to empower DC, the developers performed an activity named *choosing a cloud provider* in order to analyse three candidate public cloud platforms such as Amazon Web Services, Google App Engine, and Microsoft Azure. Each candidate platform could affect the cost, the quality of the solution architecture, and the required legacy code changes. The developers found that Google App Engine could not be a suitable candidate for DC since it did not support .NET software systems unlike Amazon AWS and Microsoft Azure that both provided such a support. After a further analysis, the developers preferred Windows Azure platform for three reasons: (i) it would require less configuration effort, (ii) it would offer faster deployment model, and (iii) developers had a consistent experience in adopting Microsoft family technologies. The concept *Choosing a cloud provider* in InformaIT conforms to the concept *choose cloud platform/provider* in the design phase of the metamodel.

In InformaIT scenario, the developers performed a task called *cloud DC architecture* indicating how the existing legacy components are mapped to the Microsoft Azure platform. For example, the legacy version of DC's database, a Microsoft SQL Server database, was replaced with SQL Azure. The metamodel generates this concept through an instantiation of concept *design cloud solution* in design phase of the metamodel.

Furthermore, the developers identified some incompatibilities between the legacy system and the cloud platform that implicated some changes to the

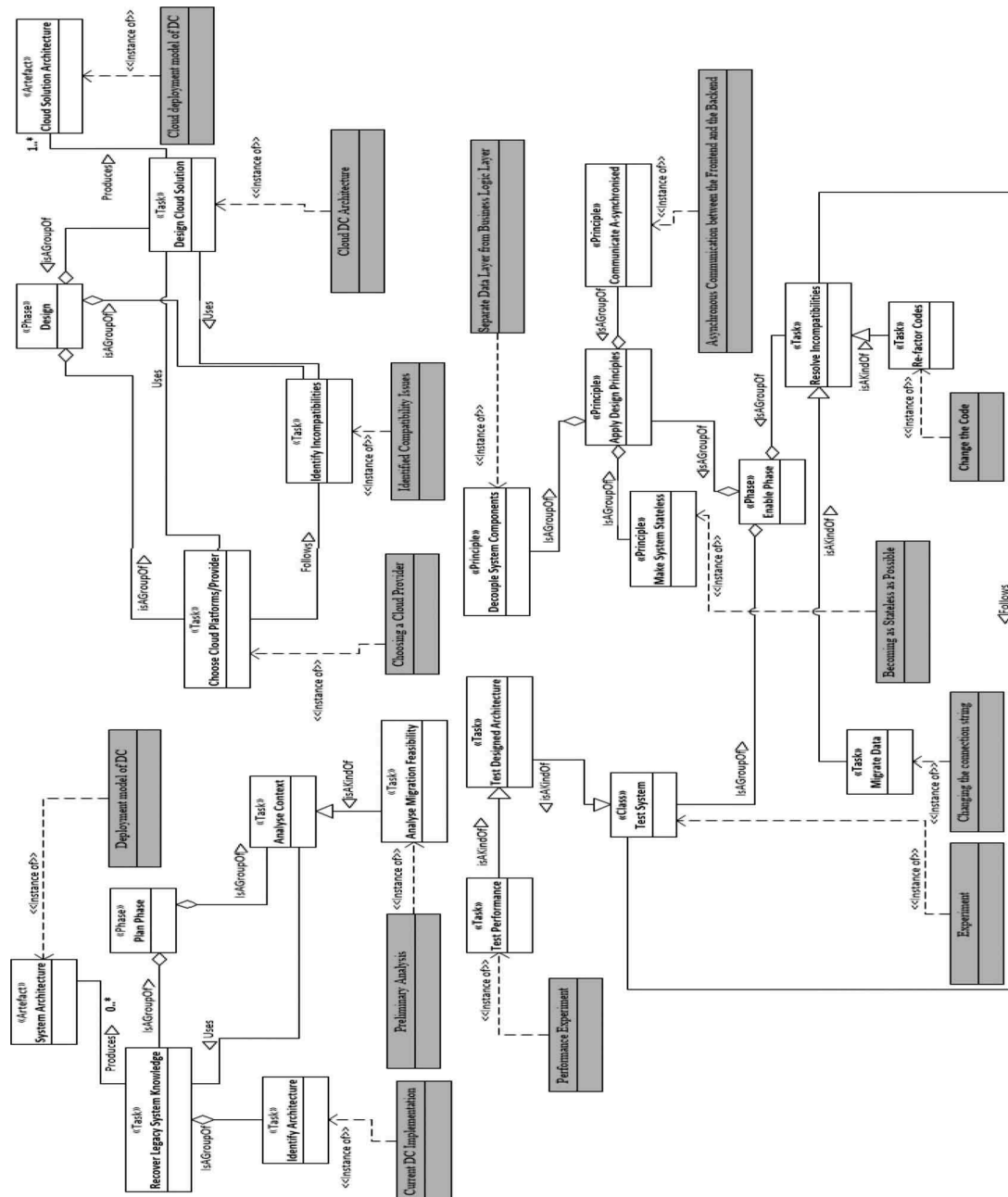


Figure 3. *InformIT* model as an instantiation of the metamodel.

current system implementation. This is referred to as *identified compatibility issues* in the metamodel. Subsequently, the migration process proceeded with some changes in DC. As an example, the data and queue storage technologies in Microsoft Azure were not compatible with regular application programming interfaces that were currently used by DC. Also, DC had been developed using Microsoft .Net 2.0 technology that was not supported by Microsoft Azure. The action to resolve this was to update DC's framework to Microsoft .Net 3.5/4. Other incompatibility issues were session management and registration of legacy components in the cloud. The classes *identify incompatibilities* and *refactor codes* in the metamodel represent the above modifications to the DC in *InformaIT* case.

We found that some changes to the DC were in the form of applying DPs defined by the concept *apply DPs* in the metamodel. For instance, DC was required to be portable between the local network and the cloud. To address this, the developers separated the data and business layers by adding a new intermediate data access layer. Hereafter, the business logic layer calls operations of the intermediate layer instead of a direct access to the data layer. In *InformaIT* project, this concept is called *separate data layer from business logic layer* which can be derived from the concept *decouple system components* as a subclass of the *apply DPs* in the metamodel. Moreover, DC stored megabytes of data per session that was a big overhead. Such a session size required more time for serialisation/de-serialisation. Developers applied a principle called *becoming as*

stateless as possible in DC architecture. This concept is an instance of the principle *make system stateless* in the metamodel.

It was likely that the performance of DC in the cloud would be decreased due to unexpected latencies occurring in cloud servers. Developers used a small Azure compute instance as well as local server instance to perform *performance experiment* to execute CPU heavy code for the document rendering. This enabled developers to compare the execution and response time of running DC in the cloud. The result of the experiment revealed potential performance bottlenecks in the cloud. The abovementioned test in this scenario conforms to the concept *test performance* in the enable phase of the metamodel.

Additionally, the suitability of the DC migration to the cloud was analysed from a cost perspective. Developers built a prototype to analyse three real-life scenarios that could describe how DC could benefit from cloud services. The cost of each migration scenario was estimated based on the pricing model of Microsoft Azure, computing instance, relational database, storage transaction, data transfer, and cache size. Building a system prototype helped developers to make a final decision regarding the cloud enablement. The concept of the prototype in this scenario is representable by *make prototype* in the metamodel. Regarding DP1, the analysing *InformaIT* confirmed some relationships between the concepts defined in the metamodel. Table 3 shows the list of relationships among the metamodel's concepts that were instantiated in this migration scenario.

Within-case analysis confirmed that majority of accommodated tasks in this scenario are derivable from the metamodel, except for a new concept *use logging* that was not covered by any concepts in the metamodel. The metamodel had a deficiency to support this concept. Since cloud environments are asynchronous, debugging and tracing a system in the cloud might be problematic (Rabetski, 2012). Applying a logging mechanism in the system architecture facilitates tracing system behaviour, resource utilisation, and identifying reasons for failures in the cloud. Therefore, the metamodel concept *apply DPs* was refined by adding a new concept named *use logging*. In Figure 2, this new concept is defined as the sub-class of *apply DPs*. It is defined as "Use the logging mechanism to facilitate the system debug and resource monitoring when running in the cloud". The inclusion of this new concept evolved the metamodel to the second version 1.1.

5.2. Cross-case analysis

Our cross-case analysis examines to the extent the metamodel adheres to the DP1 and DP2 in each scenario. Table 3 shows the collection of the process

concepts and relationships that were incorporated into the scenarios.

As for DP1, the review of the three scenarios shows that they instantiated four common metamodel concepts *recover legacy system knowledge*, *design cloud solution*, *identify incompatibilities*, and *decouple software components* in their mainstream process (see Table 3). For example, the concept *design cloud solution* in the metamodel was instantiated in three different ways in each scenario. In *InformIT*, the decision on the selection and deployment of legacy system components on cloud servers was basically a mapping between Microsoft-based legacy components and their counterparts in Microsoft Azure cloud platform. In *TOAS*, the legacy components were classified into two logical groups of platforms on the basis of similar functional behaviours. For *SpringTrader* case, components providing finance services were those selected for the migration purpose. These are an instance of *design cloud solution* defined in the metamodel.

Migration scenarios were conducted differently, and therefore, each scenario instantiated a slice of the metamodel elements to address its requirements. Except for *InformIT*, in both *TOAS* and *SpringTrader* scenarios, the activities related to handling incompatibility issues were performed (seventh row in Table 3). In *TOAS* case, developers implemented run-time adaptors to resolve incompatibilities of message formats and interfaces between the legacy system and *TOAS* cloud platform. Comparably, in *SpringTrader* case, developers implemented wrappers to separate incompatibilities between cloud microservices and the legacy system. These techniques are subsumed under the concept *develop integrators* defined in the *enable* phase of the metamodel. Furthermore, unlike the instantiation of the concept *choose cloud provider* in the *InformIT*, where developers decided to use Windows Azure cloud platform due to their former experience in using this platform, the target cloud platform in both scenarios of *TOAS* and *SpringTrader* was a pre-chosen private cloud platform. Therefore, there was no need for the instantiation of the concept *choose cloud provider* (second row in Table 3). Moreover, the case studies also confirmed the correctness of some relationships among the concepts defined in the metamodel. Finally, the second and third case studies did not result in new refinements to the metamodel.

6. Evaluation and demonstration: expert reviews

The second version of the metamodel was qualitatively examined by a panel of four domain experts with respect to all DPs. The experts are denoted by E1, E2, E3, and E4. The questionnaire form to

evaluate the metamodel is presented in [Appendix C](#). The usefulness of the metamodel was stated by the words such as “education and high-level guidance” (E1), “good communication vehicle”, and “more comprehensive list of concerns” (E2). E2 stated that “the model is clearly valuable in conveying the important concerns of a migration and how they are related. The detailed semantics help to clearly understand dependencies and possibly resulting decisions and trade-offs to be considered”. A similar opinion was expressed by E3 who said “this model can make a good impact to increase the confidence of success factor of the migration process and decrease some uncertainty. Also, this model can be used as a checklist of successful migration and this reference model makes an overall picture of migration phase and clears the roadmap for audiences to do the migration with less stress and concerns”. The advantage of the metamodel against existing models was stated by E4: “I have mostly used the classical reengineering model for legacy migration. In comparison to the model by SEI, the proposed model is more detailed in terms of underlying process and activities for migration”. In view of the DPs, here are some suggestions for the improvement of the metamodel and refinements as a consequence of each expert’s feedback.

6.1. Metamodel support for DP1 and DP2

Regarding DP1, an area of concern raised by E2 was that he believed “determining licensing issues of legacies should be made more visible in the metamodel as it can turn out to be a major task in the migration process”. In the cloud, multiple instances of a system, i.e., a virtual machine, might be created by a server based on the increased workload or rules that are triggered to scale up resources. This may cause an unintended violation of the licensing agreement that has been made between the system owner and user. This concern raised by E2 has been partially covered by concept *analyse migration cost* in the metamodel, but we have not explicated it as an individual concept in the metamodel. Utilising the knowledge source prepared during our metamodel development, a new concept named *resolve licensing issues* was added to the design phase of the metamodel ([Figure 2](#)). It is defined as follows: “Define and monitor a pay-as-you-go licensing model to handle unintended licence agreement violations due to automatic scaling”.

E3 explained that the metamodel lacks a concept called roll-back: “I have observed that migration process model should contain a concept to show rollback for the migration process”. To address this concern, we added the concept *define roll-back plan* as a subclass of *define plan* along with a new relationship in

the metamodel ([Figure 2](#)). A definition for this concept regarding the knowledge source was decided as “Define roll-back, as a plan B, to an in-house version of the legacy system in the case of occurrence of any significant risk or new application fails during the migration process. This reduces the risk and exposure to the business”. With respect to DP2, there was no major comment made by the experts. E2 suggested that relationships could be added into definitions of the concepts.

6.2. Metamodel support for DP3

The experts provided some comments related to DP3, i.e., comprehensibility of the metamodel. From E2’s viewpoint, the metamodel visualisation was unclear: “UML is not used by all stakeholders”. Likewise, E4 mentioned “a unified high-level block diagram for the reference model (unifying all those three different phases) must be presented for better illustration or reflection of the model”. As a response to the above comment, we had made a preliminary version of the metamodel using simple block diagrams. We believed if the metamodel is going to be an integral part of the model-driven development and OMG metamodeling framework (Atkinson & Kuhne, 2003), a semi-formal representation of the metamodel becomes important when the migration scale is large. In this spirit, UML is a de-facto standard for the conceptual representation in terms of organising concepts, their relationships, and decidable reasoning.

7. Discussion

7.1. Implications for research

Existing cloud migration models do not sufficiently elaborate on process components of a legacy system migration (Fahmideh et al., 2016; Jamshidi, Ahmad, & Pahl, 2013). Researchers have rather attempted to develop abstract models for cloud computing technology from different perspectives such as architectural perspective (M. Hamdaqa & Tahvildari, 2012; A. Zimmermann et al., 2013), green cloud computing requirements (Procaccianti et al., 2014), quality aspects of cloud services (Nunez et al., 2013), code refactoring and simplification Ardagna et al. (2012), and risk and compliance effort reduction (Keller & König, 2014; Martens & Teuteberg, 2011). This paper aims at alleviating problems afflicting cloud migration from a process perspective. Our metamodel supersedes existing models in a multitude of ways.

First, since the emergence of cloud computing technology, a plethora of shallow to informal models/methods has been introduced and communicated in different forms such as manual, research articles, white papers, and consulting (Fahmideh et al., 2016).

The models have different foci and concepts. This plethora of models brings benefits, accommodating the various ways on performing cloud migration. This raises difficulties for developers to access and to choose from, forcing developers to learn how the models work. The suggested metamodel of the current study advances our understanding about the cloud migration process and abstracts away from details and provides a platform-independent and unified view.

Second, the proposed metamodel provides a separation between the method design and the way the method is operationalised. Operationalisation is often bounded by underlying target cloud platforms. Separation from operationalisation issues reduces the design complexity and prevents developers from early engagement into a specific platform. This allows later them to explore each process concept in more depth. This has clear potential to improve the reusability, modularity, and maintainability of migration methods.

Finally, this research embarks on adopting a method engineering approach paving the way for hybridisation between models. Existing migration models such as Menychtas et al. (2013), Mohagheghi et al. (2010), Frey and Hasselbring (2011), Menzel and Ranjan (2012), Chauhan and Babar (2012), Strauch, Andrikopoulos, Karastoyanova, and Vukojevic (2014), and Conway and Curry (2013) largely assume that the cloud migration process is monomorphic, but none is actually a silver bullet. Each model defines a collection of activities to carry out migration process with a different scope. For instance, a model might be a better fit for moving a process-intensive and distributed workload from legacy data centres to public IaaS, whilst another model may be an adequate option to integrate legacy systems with SaaS platforms. Our metamodel provides a platform of domain concepts and their relationships grounded in what cloud computing community widely agreed. They can be selected and combined together to create a specific instance of the metamodel that fits requirements of a cloud migration project at hand. Since all of this enables a pluralist view, yet customisable and extensible, the metamodel is viewed as a configurable process model, rather than a specific model/method.

7.2. Implications for practice

Pertaining to practice, the proposed metamodel is important in two ways. First, an organisation may have its own company-wide in-house method to standardise system development but still has some deficiencies with respect to support of cloud computing concepts. The proposed metamodel can be used to augment the capability of in-house methods to carry out cloud migration projects.

Moreover, method engineers may need to select from a collection of methods that fit requirements of a given cloud migration scenario. According to Siau and Rossi (1998), metamodels are one effective way to compare family-related methods as they take place at a higher level of abstraction and capture information about methods. In the context of the cloud computing field, the proposed metamodel can act as an evaluation framework for identifying strengths, shortcomings, similarities, and differences of in-house methods. Although implied in prior studies (Fahmideh et al., 2016d), this need had not been formally addressed.

Finally, from a project management point of view, the metamodel concepts can also provide an estimation of cost and heuristics of required effort to make a legacy system cloud-enabled. Our metamodel actually responds to repeated calls by studies such as Tran, Keung, Liu, and Fekete (2011) and Quang Hieu and Asal (2012) proposing cost estimation models based on reengineering activities to be performed.

8. Limitations of the study

In this paper, the metamodel applicability has been illustrated in three idiosyncratic case studies with different characteristics. However, completely satisfying the DPs can still be subject to some arguments. This is a limitation of our research as the metamodel has been only capable in representing migration models that we examined during the case analysis as a part of our metamodeling endeavour. Appraising the metamodel in different scales of cloud migration scenarios (i.e., partial or full), and organisation size (i.e., small start-up, medium-sized organisation, and big organisations), may suggest further refinements to the metamodel concepts.

Retrospective studies present some inherent limitations (Hess, 2004). Examining the metamodel adherence to the DPs during case study analysis has relied on the accuracy of projects' databases consisting documentations, process deliverables, diagrams, and interview notes. There is a possibility of missing new concepts due to subjectivity and bias in recorded data in databases by developers. As such, there might be some concepts that could have been added to the metamodel and thus revisiting the metamodel expressivity. To alleviate this issue, we conducted follow-up communications with interviewees to confirm the validity of the secondary documents of the case studies and to provide any missing information.

With this said, although some of the refinements to the metamodel have been based on the opinions from four selected domain experts, they might have been biased and confined by their own experience and knowledge in relation to the cloud migration. Receiving feedback from a larger number of experts will reduce this threat.

9. Conclusion and future research

This study was justified by the lack of a domain language for consistently representing, sharing, and standardising knowledge related to migration to the cloud catering to specific scenarios. In addressing this gap, a generic and tunable metamodel that constitutes a reusable set of domain concepts pertinent to the cloud was proposed. We have demonstrated the suitability of the metamodel in three different case studies along with positive feedback from domain experts.

The current study points to few directions for further research. The metamodel augmentation with new concepts relevant to the post-migration, for example, continuous integration and delivery, is a possible area of improvement. Similarly, the metamodel can be extended by incorporating concepts related to the growing area of mobile cloud applications that are ran on mobile devices (Dinh, Lee, Niyato, & Wang, 2013; Giurgiu, Riva, Juric, Krivulev, & Alonso, 2009). Such applications are characterised with challenges such as battery life, bandwidth, heterogeneity, and privacy that arise in mobile environments. The UML formalism used in the proposed metamodel representation can facilitate the inclusion of new further concepts in a structured way.

The metamodel instantiation for the creation of situational methods involves some factors such as the choice of a target cloud platform, the pricing model of cloud providers, and characteristics of the development team. Making trade-offs among these factors that sometimes contradict or depend on each other has an impact on the metamodel instantiation and specialisation. In future work, one can utilise the idea of goal-driven method tailoring suggested in Cesar and Paolo (2009) and Karlsson and Ågerfalk (2011) as a baseline to define mechanisms for selecting metamodel concepts and putting them together to create method instances for a particular cloud migration scenario. We expect that this research will motivate other researchers to further explore new approaches which will systematically facilitate cloud computing adoption.

Disclosure statement

No potential conflict of interest was reported by the authors.

About the authors

Mahdi Fahmideh received a PhD degree in Information Systems from the University of New South Wales (UNSW), Sydney Australia. Mahdi's vision in research focuses on developing solutions that help organisations in adopting IT initiatives. His research outputs can be in the form of methodological approaches, frameworks, and conceptual models. He is a design science researcher focusing on

creating 'new-to-the-world' artifacts. Mahdi's research interests lie in the areas of cloud computing, big data analytics, IoT, model-driven software development, and situational method engineering. He is currently a researcher at the University of Technology Sydney (UTS). Prior to joining the academia, Mahdi was working as a programmer and system analyst in development of IT-based solutions in different industry sectors including accounting, insurance, defense, and publishing.

Farhad Daneshgar received his PhD in Information Systems from the University of Technology, Sydney Australia. He is a former senior lecturer at the UNSW and is an adjunct Professor at Bangkok University. Farhad is the creator of the Awareness Modeling Language, and has published extensively in the areas of Knowledge Management and Enterprise Systems, and was awarded twice for his Outstanding Research Article at the UNSW. Farhad is a member of editorial board in five academic journals.

Fethi Rabhi is a Professor in the School of Computer Science and Engineering at the UNSW in Australia. His main research areas are in service-oriented software engineering with a strong focus on business and financial applications. He completed a PhD in Computer Science at the University of Sheffield in 1990 and held several academic appointments in the USA and the UK before joining UNSW in 2000. He is currently actively involved in several research projects in the area of large-scale news and financial market data analysis.

Professor Ghassan Beydoun received a degree in computer science and a PhD degree in knowledge systems from the UNSW. He is currently a Professor of Information Systems at the UTS. He has authored more than 100 papers international journals and conferences. He is currently working on the metamodels for on project sponsored by Australian Research Council and Australian companies to investigate the endowing methodologies for distributed intelligent systems and supply chains with intelligence. His other research interests include multi agent systems applications, ontologies and their applications, and knowledge acquisition.

References

- Ambler, S. W. (2005). *The elements of UML (TM) 2.0 style*. Cambridge University Press.
- Antkiewicz, M., Czarnecki, K., & Stephan, M. (2009). Engineering of framework-specific modeling languages. *Software Engineering, IEEE Transactions On*, 35(6), 795–824.
- Ardagna, D., Nitto, E. D., Casale, G., et al. 2012. "ModacLOUDS: A model-driven approach for the design and execution of applications on multiple clouds," in: *Proceedings of the 4th International Workshop on Modeling in Software Engineering*. Zurich (pp. 50–56), Switzerland: IEEE Press.
- Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., Gunho, L., Patterson, D., Ariel, R., & Ion, S. (2010). A view of cloud computing. *Communications of the ACM*, 53(4), 50–58.
- Atkinson, C., & Kuhne, T. (2003). Model-driven development: A metamodeling Foundation. *Software, IEEE, 20* (5), 36–41.

- Bain, A., Mitchell, J., Sharma, R., Stefan, D., & Zimmerman, J. (2011). *A domain-specific language for computing on encrypted data (invited talk)*. LIPICs-Leibniz International Proceedings in Informatics: Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.
- Bennett, K. (1995). Legacy systems: Coping with success. *IEEE Softw*, 12(1), 19–23.
- Beydoun, G., Low, G., Henderson-Sellers, B., et al. (2009). Faml: A generic metamodel for Mas development. *Software Engineering, IEEE Transactions On*, 35(6), 841–863.
- Brandic, I., Dustdar, S., Anstett, T., Schumm, D., Leymann, F., & Konrad, R. (2010). “Compliant cloud computing (c3): Architecture and language support for user-driven compliance management in clouds,” *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*: IEEE, (pp. 244–251).
- Buyya, R., Chee Shin, Y., & Venugopal, S. (2008). “Market-oriented cloud computing: vision, hype, and reality for delivering it services as computing utilities,” *High Performance Computing and Communications, 2008. HPCC’08. 10th IEEE International Conference on*: Ieee, (pp. 5–13).
- Cameron, J. (2002). Configurable development processes. *Commun. ACM*, 45(3), 72–77.
- Cesar, G. P., & Paolo, G. (2009). “Method Construction by Goal Analysis,”
- Chauhan, M. A., & Babar, M. A. (2012). “Towards process support for migrating applications to cloud computing,” *Cloud and Service Computing (CSC), 2012 International Conference on*, pp. 80–87.
- Chow, R., Golle, P., Jakobsson, M., Shi, E., Staddon, J., Masuoka, R., & Molina, J. (2009). “Controlling data in the cloud: Outsourcing computation without outsourcing control,” *Proceedings of the 2009 ACM workshop on Cloud computing security* (pp. 85–90): ACM.
- Cimato, S., Damiani, E., Zavatarelli, F., & Menicocci, R. (2013). Towards the certification of cloud services. *Services (SERVICES), 203 IEEE Ninth World Congress On: IEEE*, 92–97.
- Clark, W. (1986). *Human migration*. Beverly Hills: Sage. Google Scholar.
- Conway, G., & Curry, E. (2013). The IVI cloud computing life cycle. In *Cloud computing and services science* (pp. 183–199). Springer.
- Dinh, H. T., Lee, C., Niyato, D., & Wang, P. (2013). A survey of mobile cloud computing: Architecture, applications, and approaches. *Wireless Communications and Mobile Computing*, 13(18), 1587–1611.
- Dougherty, B., White, J., & Schmidt, D. C. (2012). Model-driven auto-scaling of green cloud computing infrastructure. *Future Generation Computer Systems*, 28(2), 371–378.
- Erlikh, L. (2000). Leveraging legacy system dollars for E-business. *IT Professional*, 2(3), 17–23.
- Fahmideh, M., Daneshgar, F., Beydoun, G., & Rabhi, F. (2017a). Challenges in migrating legacy software systems to the cloud—An empirical study. *Information Systems*, 67(SupplementC), 100–113. 2017/07/01/.
- Fahmideh, M., Daneshgar, F., Beydoun, G., & Rabhi, F. (2017b). “Key challenges during legacy software system migration to cloud computing platforms—An empirical study,”
- Fahmideh, M., Daneshgar, F., Low, G., & Beydoun, G. (2016). Cloud migration process—A survey, evaluation framework, and open challenges. *Journal of Systems and Software*, (120), 31–69.
- Fahmideh, M., Daneshgar, F., & Rabhi, F. (2016a). “Cloud computing adoption: An effective tailoring approach.” 27th Australasian Conference on Information Systems (ACIS).
- Fehling, C., Leymann, F., Rüschlin, J., & Schumm, D. (2012). Pattern-based development and management of cloud applications. *Future Internet*, 4(1), 110–141.
- Fehling, C., & Retter, R. (2011). “Cloud computing patterns.” <http://cloudcomputingpatterns.org>.
- Frey, S., & Hasselbring, W. (2011). The Cloudmig approach: Model-based migration of software systems to cloud-optimized applications. *International Journal on Advances in Software*, 4:3(4), 342–353.
- Gholami, M. F., Daneshgar, F., & Rabhi, F. (2016). “Cloud migration: Methodologies: preliminary findings.” European Conference on Service-Oriented and Cloud Computing—CloudWays 2016 Workshop.
- Giurgiu, I., Riva, O., Juric, D., Krivulev, I., & Alonso, G. (2009). Calling the cloud: Enabling mobile phones as interfaces to cloud applications. In *Middleware, 2009* (Springer), 83–102.
- Gonzalez-Perez, C., & Henderson-Sellers, B. (2008). *Metamodelling for software engineering*. Wiley Publishing.
- Gregor, S., & Hevner, A. R. (2013). Positioning and presenting design science research for maximum impact. *MIS Quarterly*, 37(2), 337–356.
- Hamdaqa, M., Livogiannis, T., & Tahvildari, L. (2011). A reference model for developing cloud applications. In: *Proceedings of CLOSER, 2011*, 98–103.
- Hamdaqa, M., & Tahvildari, L. (2012). “Cloud computing uncovered: A research landscape,” *Advances in Computers*, (86), 41–85.
- Henderson-Sellers, B., & Gonzalez-Perez, C. (2010). Granularity in conceptual modelling: Application to metamodels. In J. Parsons, M. Saeki, P. Shoval, C. Woo, & W. Y. (Eds.), *Conceptual modeling—Er 2010: 29th International Conference on Conceptual Modeling, Vancouver, BC, Canada, November 1-4, 2010. Proceedings* (pp. 219–232). Berlin, Heidelberg: Springer Berlin Heidelberg.
- Hess, D. R. (2004). Retrospective studies and chart reviews. *Respiratory Care*, 49(10), 1171–1174.
- Hollenbeck, J. R. (2008). *The role of editing in knowledge development: consensus shifting and consensus creation* (pp. 16–26). Springer.
- Isard, M., & Yu, Y. (2009). “Distributed data-parallel computing using a high-level programming language,” *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*: ACM, pp. 987–994.
- Jamshidi, P., Ahmad, A., & Pahl, C., (2013). “Cloud migration research: A systematic review,” *Cloud Computing, IEEE Transactions on* 1(2), 142–157.
- Jonkers, H., Stroucken, M., & Vdovjak, R. (2006). Bootstrapping domain-specific model-driven software development within Philips. In *6th OOPSLA Workshop on Domain Specific Modeling (DSM 2006)* (pp. 10). Citeseer.
- Karlsson, F., & Ågerfalk, P. J. (2011). Towards structured flexibility in information systems development: Devising. In *Theoretical and practical advances in information systems development: Emerging trends and approaches* (pp. 214).
- Keller, R., & König, C. (2014). “A reference model to support risk identification in cloud networks”.
- Khajeh-Hosseini, A., Greenwood, D., Smith, J. W., & Sommerville, I. (2012). The cloud adoption toolkit:

- Supporting cloud adoption decisions in the enterprise. *Software: Practice and Experience*, 42(4), 447–465.
- Khajeh-Hosseini, A., Greenwood, D., & Sommerville, I. (2010, July). "Cloud migration: A case study of migrating an enterprise it system to IaaS," *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*, (pp. 450–457).
- Kitchenham, B., Pearl Brereton, O., Budgen, D., Turner, M., Bailey, J., & Linkman, S. (2009). Systematic literature reviews in software engineering—A systematic literature review. *Information and Software Technology*, 51(1), 7–15. doi:10.1016/j.infsof.2008.09.009
- Koçak, S. A., Miranskyy, A., Alptekin, G. I., Bener, A. B., & Cialini, E. (2013). The impact of improving software functionality on environmental sustainability. In *On information and communication technologies* (pp. 95).
- La, H. J., & Kim, S. D. (2009). A systematic process for developing high quality SaaS cloud services. In *Cloud Computing* (pp. 278–289). Springer.
- Lee, E. S. (1966). A theory of migration. *Demography*, 3(1), 47–57. doi:10.2307/2060063
- Leymann, F. (2011). Cloud computing. *it-Information Technology Methoden Und Innovative Anwendungen Der Informatik Und Informationstechnik*, 53(4), 163–164.
- Lindland, O. I., Sindre, G., & Solvberg, A. (1994). "Understanding quality. *Software*," *IEEE Modeling*, 11 (2), 42–49.
- Linthicum, D. (2012). "Why cloud computing projects fail?" Available at: <http://www.slideshare.net/Linthicum/why-cloud-computing-projects-fail>, last access October 2016).
- Liu, F., Tong, J., Mao, J., Bohn, R., Messina, J., Badger, L., & Leaf, D. (2011). NIST cloud computing reference architecture. *NIST Special Publication*. (500), 292.
- Loutas, N., Kamateri, E., Bosi, F., & Tarabanis, K. (2011). "Cloud computing interoperability: The state of play," *Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on*: IEEE, (pp. 752–757).
- Martens, B., & Teuteberg, F. (2011). "Risk and compliance management for cloud computing services: Designing a reference model,"
- Menychtas, A., Santzaridou, C., Kousiouris, G., Varvarigou, T., Orue-Echevarria, L., Alonso, Gorrionogitia, J., Brunelière, H., Strauss, O., & Senkova, T. (2013, September). "Artist methodology and framework: A novel approach for the migration of legacy software on the cloud," In *Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), 2013 15th International Symposium on*: IEEE (pp. 424–431).
- Menzel, M., & Ranjan, R. (2012). "Cloudgenius: Decision support for web server cloud migration," *Proceedings of the 21st international conference on World Wide Web*: (pp. 979–988). ACM.
- Misra, S. C., & Mondal, A. (2011). Identification of a company's suitability for the adoption of cloud computing and modelling its corresponding return on investment. *Mathematical and Computer Modelling*, 53(3), 504–521.
- Mohagheghi, P., Berre, A., Henry, A., Barbier, F., & Sadovykh, A. (2010). Remics-reuse and migration of legacy applications to interoperable cloud services. In E. Nitto & R. Yahyapour (Eds.), *Towards a service-based internet* (pp. 195–196). Springer Berlin Heidelberg.
- Nowak, A., Breitenbücher, U., & Leymann, F. (2014). "Automating green patterns to compensate Co2 emissions of cloud-based business processes," *ADVCOMP 2014, The Eighth International Conference on Advanced Engineering Computing and Applications in Sciences*, (pp. 132–139).
- Nunez, D., Fernandez-Gago, C., Pearson, S., & Felici, M. (2013). "A metamodel for measuring accountability attributes in the cloud," *Cloud Computing Technology and Science (CloudCom), 2013 IEEE 5th International Conference on*: IEEE, (pp. 355–362).
- Oliveira, T., Thomas, M., & Espadanal, M. (2014). Assessing the determinants of cloud computing adoption: An analysis of the manufacturing and services sectors. *Information & Management*, 51(5), 497–510.
- Othman, S. H., & Beydoun, G. (2013). Model-driven disaster management. *Information & Management*, 50(5), 218–228.
- Peppers, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S. (2008). A design science research methodology for information systems research. *Journal of Management Information Systems*, 24(3), 45–77.
- Pepitone, J. (2011). Amazon EC2 outage downs Reddit, Quora. In *Retrieved May* (Vol. 17, pp. 2011).
- Procaccianti, G., Lago, P., & Lewis, G. A. (2014). "Green architectural tactics for the cloud,"
- Quang Hieu, V., & Asal, R. (2012). Legacy application migration to the cloud: Practicability and methodology. In *Services (SERVICES), 2012* (pp. 270–277). IEEE Eighth World Congress on.
- Rabetski, P. (2012). "Migration of an on-premise application to the cloud,"
- Ranabahu, A. H., Maximilien, E. M., Sheth, A. P., & Thirunarayan, K. (2011). "A domain specific language for enterprise grade cloud-mobile hybrid applications," *Proceedings of the compilation of the co-located workshops on DSM'11, TMC'11, AGERE'11, AOOPE'11, NEAT'11, & VMIL'11* (pp. 77–84). ACM.
- Ried, S., Kisker, H., Matzke, P., Bartels, A., & Lisserman, M. (2011). Sizing the cloud, understanding and quantifying the future of cloud computing. In *Forrester research, inc*, 21, 13.
- Rossi, M., & Brinkkemper, S. (1996). Complexity metrics for systems development methods and techniques. *Information Systems*, 21(2), 209–227.
- Rossi, M., Ramesh, B., Lyytinen, K., & Tolvanen, J. P. (2004). Managing evolutionary method engineering by method rationale. *Journal of the Association for Information Systems*, 5(9), 356–391.
- Siau, K., & Rossi, M. (1998). "Evaluation of information modeling methods—A review," *System Sciences, 1998., Proceedings of the Thirty-First Hawaii International Conference on*: IEEE, (pp. 314–322).
- Sledziewski, K., Bordbar, B., & Anane, R. (2010). "A DSL-based approach to software development and deployment on cloud," *Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on*: IEEE, (pp. 414–421).
- Sneed, H. M. (1995). Planning the reengineering of legacy systems. *Software, IEEE*, 12(1), 24–34.
- Strauch, S., Andrikopoulos, V., Karastoyanova, D., & Vukojevic, K. (2014). "Migrating Escience Applications to the Cloud: Methodology and Evaluation,"
- Toosi, A. N., Calheiros, R. N., & Buyya, R. (2014). Interconnected cloud computing environments: Challenges, taxonomy, and survey. *ACM Computing Surveys (CSUR)*, 47(1), 7.
- Tran, V., Keung, J., Liu, A., & Fekete, A. (2011). "Application migration to cloud: A taxonomy of critical factors,"

- Proceedings of the 2nd International Workshop on Software Engineering for Cloud Computing* (pp. 22–28). ACM.
- Truong, H.-L., & Dustdar, S. (2010). Composable cost estimation and monitoring for computational applications in cloud computing environments. *Procedia Computer Science*, 1(1), 2175–2184.
- Tsidulko, J. (2016). “The 10 biggest cloud outages of 2016,” Available at <http://www.crn.com/slide-shows/cloud/300081477/the-10-biggest-cloud-outages-of-2016-so-far.htm>.
- UML, O. (2004). 2.0 superstructure specification. In *OMG, Needham*.
- Venters, W., & Whitley, E. A. (2012). A critical review of cloud computing: Researching desires and realities. *Journal of Information Technology*, 27(3), 179–197.
- Weimer, M., Condie, T., & Ramakrishnan, R. (2011). Machine learning in Scalops, a higher order cloud computing language. *NIPS 2011 Workshop on Parallel and Large-Scale Machine Learning (Biglearn)*, 9, 389–396.
- Wettinger, J., Behrendt, M., Binz, T., et al. (2013). Integrating configuration management with model-driven cloud management based on Tosca. *CLOSER* (pp. 437–446).
- Whetten, D. A. (1989). What Constitutes a Theoretical Contribution?. *Academy of Management Review*, 14(4), 490–495. doi:10.5465/amr.1989.4308371
- Yang, H., & Tate, M. (2012). A descriptive literature review and classification of cloud computing research. *Communications of the Association for Information Systems*, 31(2), 35–60.
- Zech, P., Felderer, M., & Breu, R. (2012). “Cloud risk analysis by textual models,” in: *Proceedings of the 1st International Workshop on Model-Driven Engineering for High Performance and CCloud computing*. (pp. 1–6), Austria: ACM.
- Zhang, L.-J., & Zhou, Q. (2009). “Ccoa: Cloud computing open architecture,” *Web Services, 2009. ICWS 2009. IEEE International Conference on: Ieee*, (pp. 607–616).
- Zimmermann, A., Pretz, M., Zimmermann, G., Firesmith, D. G., & Petrov, I. (2013). “Towards service-oriented enterprise architectures for big data applications in the cloud,” *Enterprise Distributed Object Computing Conference Workshops (EDOCW), 2013 17th IEEE International: IEEE*, (pp. 130–135).
- Zimmermann, O., Mikšovic, C., & Küster, J. M. (2012). “Reference architecture, metamodel, and modeling principles for architectural knowledge management. *Journal of Systems and Software*, 85(9), 2014–2033.

Appendix A

The list of studies that have been identified through a systematic literature review and used in the metamodel development.

Authors and title	Constructs in the Study	Total
Krasteva, I., S. Stavru, et al., "Agile model-driven modernisation to the service cloud"	1) Requirements and Feasibility Activity Area, 2) Recover Activity Area, 3) Migrate Activity Area, 4) Validation Activity Area, 5) Supervise Activity Area, 6) Interoperability Activity Area, 7) Continuous Integration, 8) Identify additional requirements and specify new requirements in UML diagrams	8
Beserra, P. V., A. Camara, et al., "Cloudstep: A step-by-step decision process to support legacy application migration to the cloud"	1) Define Organisation Profile, 2) Evaluate Organisational Constraints, 3) Define Application Profile, 4) Define Cloud Provider Profile, 5) Evaluate Technical and/or Financial Constraints, 6) Change Cloud Provider, 7) Define Migration Strategy	7
Mohagheghi, P., "Software engineering challenges for migration to the service cloud paradigm: Ongoing work in the REMICS project"	1) Establishing the Context, 2) Modernising the Software Architecture, 3) Modernising Data, 4) Managing Non-Functional and QoS Requirements in the Cloud, 5) Verification and Validation in the Cloud, 6) Verifying the Scalability of Applications	6
Conway, G. and E. Curry, "The IVI cloud computing life cycle"	1) Investigate, 2) Identify, 3) Implementation Strategy, 4) Business Design, 5) Select, 6) Negotiate, 7) Roll Back, 8) Manage the Supply Chain, 9) Review	9
Khajeh-Hosseini, A., D. Greenwood, et al., "Cloud migration: A case study of migrating an enterprise it system to laas"	1) Stakeholder Impact Analysis, 2) Identifying Changes, 3) Identifying Change Consequences on the Context	3
Tran, V., J. Keung, et al., "Application migration to cloud: A taxonomy of critical factors"	1) Training or Learning Curve, 2) Installation and Configuration, 3) Code Modification, 4) Migration, 5) Testing	5
Khajeh-Hosseini, A., D. Greenwood, et al., "The cloud adoption toolkit: Supporting cloud adoption decisions in the enterprise"	1) Technology Suitability Analysis, 2) Stakeholder Impact Analysis, 3) Organisational Change	3
Kundra, V., "Federal cloud computing strategy"	1) Determine Cloud Readiness, 2) Decommissioning Legacy Assets, 3) Build New Skill Sets, 4) Actively Monitor SLAs, 5) Selecting Services to Move to the Cloud, 6) Contract Effectively, 7) Integrate Services	7
Chauhan, M. A. and M. A. Babar, "Towards process support for migrating applications to cloud computing"	1) Requirements Identification, 2) Identification of Potential Cloud Hosting Environments, 3) Identification of Potential Architecture Solutions, 4) Evaluation of Cloud Environments for Cloud Specific Quality Attributes, 5) Analysing Application Compatibility With Potential Cloud Environments, 6) Evaluation of Proposed Solutions and Effectuated Components Against Target Platforms, 7) Implementation and Refactoring, 8) System Requirements, 9) Finalised Design Decision and Modified System Architecture, 10) List of Potential Cloud Environments, 11) Selected Cloud Environments, 12) Trade-off Analysis of Cloud Environment, 13) Trade-off Analysis of Quality Attributes Cloud, 14) Implementation	14
Strauch, S., et al., "Migrating eScience applications to the cloud: Methodology and evaluation"	1) Migration Strategy, 2) Describe Desired Cloud Data Hosting Solution, 3) Select Cloud Data Store or Data Service, 4) Identify Patterns to Solve Potential Migration Conflicts, 5) Refactor Application Architecture, 6) Migrate Data, 7) Network Adaptation	7
S. Strauch, V. A., D. Karastoyanova, F. Leymann, "Migrating enterprise applications to the cloud: Methodology and evaluation"	1) Select Migration Scenario, 2) Describe Desired Cloud Data Hosting Solution, 3) Select Cloud Data Store or Data Service, 4) Identify Patterns to Solve Potential Migration Conflicts, 5) Refactor Application Architecture, 6) Migrate Data	6
Leymann, F., et al., "Moving applications to the cloud: An approach based on application model enrichment"	1) Architecture Model of the Application to be Moved to the Cloud, 2) Cloud Distribution, 3) Executable or Virtual Images, 4) Variability Points	4
Zhang, W., et al., "Migrating legacy applications to the service cloud"	1) Architectural Representation of the Legacy Application, 2) Redesign the Architecture Model, 3) MDA Transformation, 4) Web Service Generation, 5) Web Service-Based Invocation of legacy Functionalities, 6) Selection of Suitable Cloud Computing Platform, 7) Web Service Deployment in the Service Cloud	7
Frey, S., et al., "Automatic conformance checking for migrating software systems to cloud infrastructures and platforms"	1) Extraction, 2) Selection, 3) Generation, 4) Adaptation, 5) Evaluation, 6) Transformation	6
Miranda, J., et al., "Assisting cloud service migration using Software adaptation techniques"	1) Identifying Mismatch Points, 2) Solving Mismatches, 3) Adapter Generation and Injection	3
Fehling, C., et al., "Service migration patterns—decision support and best practices for the migration of existing service-based applications to cloud environments"	1) Temporarily Unavailable, 2) Stateless Component Swapping, 3) Migration Target	3
Tak, B. C., et al., "To move or not to move: The economics of cloud computing"	1) Application Hosting Choices, 2) Licensing Structures, 3) Application Partitioning	3
Guillén, J., et al., "A service-oriented framework for developing cross cloud migratable software"	1) Source Code Transformation, 2) Service Adapters, 3) Stateless and Stateful Services, 4) Cloud Variability Model, 5) Cloud Deployment Plan, 6) Heterogeneous Test Environment	6
Ardagna, D., et al., "ModacLOUDS: A model-driven approach for the design and execution of applications on multiple clouds"	1) Dynamic Re-Deployment, 2) Cloud Selection, 3) Adaptation Actions, 4) Replicating	4
Hajjat, M., et al., "Cloudward bound: Planning for beneficial migration of enterprise applications to the cloud"	1) Migrating Reachability Policies, 2) Replication of Storage in the Local and Cloud Data-Centres, 3) Filter Unauthorised Packets	3
Moen, H., et al., "Design and evaluation of a hierarchical application placement algorithm in large scale clouds"	1) Application Placement, 2) Decoupling	2
SUN, K., Li, Y., "Effort estimation in cloud migration process"	1) Migration Planning, 2) Requirement Collection, 3) Discovery, 4) Assessment, 5) Map, 6) Provision, 7) Migrate, 8) Test	8

(Continued)

(Continued).

Authors and title	Constructs in the Study	Total
Rabetski, P. and G. Schneider, "Migration of an on-premise application to the cloud: Experience report"	1) Distributed Deployment Model, 2) Choosing a Cloud Provider, 3) Cloud DC Architecture, 4) Identified Compatibility Issues, 5) Separate Data Layer from Business Logic Layer, 6) Become as Stateless as Possible	6
C., Pahl, H. Xiong, et al., "A comparison of on-premise to cloud migration approaches"	1) Decommissioning of Old Infrastructure, 2) Cloud Architecture Solution, 3) Backup Systems, 4) SLA Negotiations, 5) Post-Usage Billing, 6) Necessary Skills, 7) Statelessness, 8) Testing and Monitoring, 9) Licence Model, 10) Implement Data Migration, 11) Data Protection	11
Laszewski, T. and P. Nauduri, "Migrating to the cloud: Oracle client/server modernisation"	1) Assessment, 2) Migration Service Provider, 3) Migration Effort Estimate, 4) Training Requirements, 5) IT Resource Requirement for the Migration Project, 6) Capture Exhaustive Amounts of Information from the Source Systems, 7) Database Schema Layout, 8) Testing, 9) Optimisation, 10) Deployment	10
Guo, X., "Evaluation of a methodology for migration of the database layer to the cloud based on an eScience case study"	1) Select Migration Scenario, 2) Describe Desired Cloud Data Hosting Solution, 3) Select Cloud Data Store or Data Service, 4) Identify Patterns to Solve Potential Migration Conflicts, 5) Refactor Application Architecture, 6) Migration Data, 7) Test	7
Ahmad, Aakash, and Muhammad Ali Babar. "A framework for architecture-driven migration of legacy systems to cloud-enabled software"	1) Feasibility Study, 2) Requirement Analysis, 3) Decision on Cloud Providers, 4) Migration Strategy Development, 5) Architecture Recovery and Consistency, 6) Legacy Architecture Description, 7) Architectural Change Implementation, 8) Code Consistency Conformance, 9) Migration Planning, 10) Migration Plan, 11) Cloud-Service Architecture, 12) Legacy Architecture	12
Ridha, Gadhgadhi, Khazri Saida, and Cheriet Mohamed. "OPENICRA: Towards a generic model for automatic deployment of applications in the cloud computing"	1) Provision of Computing Resources, 2) Deploy Distributed Applications, 3) Synchronisation (Local parts and those in the Cloud), 4) Isolation, 5) Stateful and Stateless Modes, 6) Templates	6
Council, C. S. C., "Migration applications to public cloud services: Roadmap for success"	1) Gather Technical Discovery Data, 2) Document Architecture Details, 3) Deploy the Cloud Environment, 4) Update Documentation, 5) Test Network and Server Connectivity, 6) Instal and Configure the Application, 7) Enable Licence Tracking, 8) Implement Integrations, 9) Configure Monitoring, 10) Harden Production Environment, 11) Communicate Changes to Users, 12) Post Migration Checkpoint, 13) Migration Go/No Go Meeting, 14) Adapt a Flexible Integration Model, 15) Document Migration Duration, 16) Notify Users of Outage, 17) Configure Database Backups, 18) Cost Analysis, 19) Determine Integrations	19
Menzel, M., "(MC2) 2: A generic decision-making framework and its application to cloud computing"	1) Define Scenario, 2) Define Alternatives, 3) Define Requirements, 4) Choose Multi-Criteria Decision-Making Method	4
Quang Hieu, V. and R. Asal, "Legacy application migration to the cloud: Practicability and methodology"	1) Compatibility Checklist (Programming language compatibility, Third-Party Library Compatibility, and GUI Compatibility), 2) Database Compatibility, 3) Migration Cost Estimation	3
H.A.Huru, "MILAS: Modernising Legacy Applications towards Service Oriented Architecture (SOA) and Software as a Service (SaaS)"	1) Recover to Architecture Representation (Analysis of Legacy Architecture, Retrieving New and Old Requirements), 2) Select Strategy, 3) Gradually Refactor Code, 4) Performance Isolation and Scalability, 5) Availability Isolation, 6) On-the-fly Customisation, 7) Select Target Platform, 8) Configure the Running Environments	8
Benguria, G., Elvesæter, B., Ilieva, S., "REuse and Migration of legacy applications to Interoperable Cloud Services"	1) Requirements Phase, 2) Recover Phase, 3) Migrate Phase, 4) Validate Phase, 5) Supervise Phase, 6) Withdrawal Phase	6
Tang, K., Zhang J.M, Feng, C.H, "Application centric lifecycle framework in cloud"	1) Design the Functionality and Fulfil non-Function Requirements, 2) Deployment Model, 3) Auto-Scaling, 4) Update Patch Management, 5) Health Checking, 6) Withdraw, 7) Server Bound	7
Bezemer, C.-P., et al., "Enabling multi-tenancy: An industrial experience report"	1) Authentication, 2) Configuration, 3) Database	3
Guo, C. J., et al., "A framework for native multi-tenancy application development and management"	1) Security Isolation, 2) Performance Isolation, 3) On-the-Fly Customisation, 4) Availability Isolation	4
Mietzner, R., et al., "Cafe: A generic configurable customisable composite cloud application framework"	1) Application Model, 2) Deployment of Components	2
Strauch, S., Andrikopoulos, V., Gómez Sáez, S., Leymann, F. "Transparent access to relational databases in the cloud using a multi-tenant ESB"	1) Management and Configuration, 2) On-premise and Off-premise Support, 3) Transparent Data Access, 4) Loose Coupling, 5) Security, 6) Backward Compatibility & Extensibility, 7) Tenant-based Identification, 8) Customisable Interfaces, 9) Performance Isolation	9
Marimuthu C, K. Chandra Sekaran, "Software development for cloud: An experiential study"	1) Multi-Tenancy, 2) Security, 3) Easy Customisation, 4) Integration with Legacy Applications, 5) Configuration & Site Building, 6) Continuous Updates	6
Anstett, T., et al., "Towards BPEL in the cloud: Exploiting different delivery models for the execution of business processes"	1) Access Control and Isolation, 2) Obfuscated Processes, 3) Allowing Users to Customise Business	3
Varia, J., "Architecting for the cloud: Best practices"	1) Implement Elasticity, 2) Decoupling Components, 3) Asynchronous Components, 4) Making Applications Stateless, 5) Protect Data in Transit, 6) Encrypt the Data, 7) Regularly Download Patches, 8) Monitor	8
Bessani, A., et al., "DepSky: Dependable and secure storage in a cloud-of-clouds"	1) Encryption of Data, 2) Data Replication on Diverse Clouds	2
Curino, C., et al., "Relational cloud: A database-as-a-service for the cloud"	1) Run Over Encrypted Data, 2) Efficient Multi-Tenancy, 3) Elastic Scalability, 4) Database Partitioning Model, 5) Database Partitioning	5
Binz, Leymann et al., "CMotion: A framework for migration of applications into and between clouds"	1) Runtime Adaptor, 2) Application Model, 3) Generate and Combine Alternatives, 4) Selection and Evaluation, 5) Deployment	5

(Continued)

(Continued).

Authors and title	Constructs in the Study	Total
Fehling, C., et al., "Pattern-based development and management of cloud applications"	1) Management Planning, 2) Replicate and Synchronise Data, 3) Application Variability, 4) Enable Asynchronous Communication, 5) Elasticity Management (Elastic Component, Elastic Queue, Elastic Load Balancer)	5
Chauhan, M. A. and M. A. Babar, "Migrating service-oriented system to cloud computing: an experience report"	1) Message Level Security, 2) Encrypted Storage, 3) Secure Communication Exchange	3
Zagarese, Q., et al., "Enabling advanced loading strategies for data intensive web services"	1) Designing Messages Dynamic Granularity, 2) Decoupling	2
Cisco Systems, "Planning the migration of enterprise applications to the cloud"	1) Architecture Requirements, 2) Geographic Access Requirements, 3) Application Dependency Mappings, 4) Application Profiling, 5) Application Migration Plan and Tests, 6) Application Architectures	6
Duplyakin, D., et al., "Rebalancing in a multi-cloud environment"	1) Rebalancing Policies (Upscaling, Downscaling, Multi-Cloud Request), 2) Resource Provisioning	2
Ali Babar, M., Chauhan, M.A., "A tale of migration to cloud computing for sharing experiences and observations"	1) Requirements Identification, 2) Requirements Analysis for Architecture Decisions, 3) Modified Architecture, 4) Testing	4
Pahl, C. and H. Xiong, "Migration to PaaS clouds-migration process and architectural concerns"	1) Component Decommissioning, 2) Statelessness, 3) Up-front Licensing, 4) Scale-out and Resource Efficiency, 5) Skills Shortages, 6) Systems Integrator, 7) Business Analysis, 8) Architecture Design, 9) Mapping and Moving	9
Saleh, E., Shaabani, N., Meinel, C., "A framework for migrating traditional web applications into multi-tenant SaaS"	1) Billing Model and Pricing, 2) Monitoring Resource Utilisation	2
Karampaglis, Z., Mentis, A., Rafailidis, F., Tsolakidis, P., Ampatzoglou, A., "Secure migration of Legacy applications to the web"	1) Monitor Cloud Resources, 2) Monitor Application Components	2
Ilieva, S., Krasteva, I., Benguria, G., Elvesæter, B., "Enhance your model-driven modernisation process with agile practices"	1) Requirements and Feasibility, 2) Recover, 3) Migrate, 4) Validate, 5) Control and Supervise, 6) Withdrawal	6
Bahga, A., Madiseti, V.K., "Rapid prototyping of multitier cloud-based services and systems"	1) Component Design, 2) Architecture Design 3) Loose Coupling, 4) Asynchronous Communication, 5) Stateless Design, 6) Load-Balancing, 7) Scaling, 8) Deployment Design, 9) Database Replication Strategies, 10) Testing	10
Logicalis, "Logicalis, Migrating to the cloud—A logicalis how to guide"	1) Compute Requirements, 2) Application Identification, 3) Disaster Recovery), 4) Network Design, 5) Pilot and Commissioning, 6) Testing Requirements, 7) A Readiness Assessment	7
Lindner, M.A., McDonald, F., Conway, G., Curry E., "Understanding cloud requirements – A supply chain lifecycle approach"	1) Investigate, 2) Identify, 3) Implementation Strategy, 4) Business Design, 5) Selection, 6) Negotiation and Sign-off, 7) Management of the Supply Chain, 8) Review	8
Zhu.Y., "A Platform for Changing Legacy Application to Multi-tenant Model"	1) Tenant Management, 2) Billing management, 3) Data Recovery, 4) Resource Monitor, 5) Database Transformation, 6) Tenant Template, 7) Backup and Restore, 8) Platform Transformation, 9) Tenant Data Isolation Test, 10) Migration Platform Performance Testing	10
Zhou, H., Yang, H., Hugill, A., "An ontology-based approach to reengineering enterprise software for cloud computing"	1) Preparation, 2) Capture (Diagram Extraction Unit, Ontology Extraction Unit, Model Transformation Unit), 3) Coding (Source Code Ontology, Database Ontology, Coding and Capture), 4) Integration, 5) Deployment	5
Frey.S., Hasselbring, W., "An extensible architecture for detecting violations of a cloud environment's constraints during legacy software system migration"	1) Extraction (Extraction of Legacy Architecture), 2) Selection, 3) Generation (Produces Target Architecture and a Mapping Model), 4) Adaptation (Adjust the Target Architecture), 5) Evaluation (Runtime Simulation), 6) Transformation (Generating Target Architecture), 7) Violation Severity	7
Frey, S. and W. Hasselbring, "The cloudmig approach: Model-based migration of software systems to cloud-optimised applications"	1) Extraction, 2) Selection, 3) Generation (Model Transformation, Configuration), 4) Adaptation, 5) Evaluation, 6) Target Architecture, 7) Rating, 8) Cloud Incompatible	8
Banerjee, J., "Moving to the cloud: Workload migration techniques and approaches"	1) Initial Screening and Analysis, 2) Conducting Requirement Workshops, 3) Planning, 4) Prepare the Target, 5) De-Commissioned, 6) Tuning the Target Environment (Updated with the new IP Address), 7) Back on Premise 8) Security Patches, 9) Security Policy, 10) Migration Plan, 11) Final Testing and Go-Live	11
Nussbaumer, N. and X. Liu, "Cloud migration for SMEs in a service oriented approach"	1) Service Candidates, 2) Platform Selection, 3) Service Billing, 4) Disaster Recovery, 5) Load Balancing, 6) Data Protection, 7) Redundancy, 8) Customisation), 9) Deployment, 10) Application Deployment Model, 11) Service Management & Operation	11
Menychtas, A., Santzaridou, C., Kousiouris, G., Varvarigou, T., "ARTIST methodology and framework: A novel approach for the migration of legacy software on the cloud"	1) Technical Feasibility Analysis, 2) Business Feasibility Analysis, 3) Application Discovery and Understanding, 4) Target Environment Specification, 5) Detect Inconsistencies, 6) Monitoring, 7) Billing	7
Andrikopoulos, V., Binz, T., Leymann, F., Strauch, S., "How to adapt applications for the cloud environment"	1) Data Store Functionality Extension, 2) Encrypted Data, 3) Synchronisation Mechanism, 4) Reconfigure Addressing, 5) Granularity of Interactions, 6) Analysing Application Dependencies, 7) Tenant Isolation, 8) Pricing Models, 9) Encrypted Communications, 10) SLA Monitoring, 11) Elasticity Mechanisms, 12) Selection of Cloud Providers, 13) Cloud Deployment Models, 14) Provider Incompatibilities, 15) Software Licensing, 16) Cost of Migration and Operation, 17) Decouple BL from DBL, 18) Emulator, 19) Network Performance	19
S., Rajaraajeswari, R., Pethuru, "Cloud application modernisation and migration methodology"	1) Evaluate, 2) Evaluation of Cloud Environment, 3) Application Architectures, 4) Scale-Up and Scale-Out Architectures, 5) Application Profiling, 6) Optimise, 7) Operate, 8) Training	8
Duarte and Silva 2013, "Cloud maturity model"	1) Investigate, 2) Target, 3) Strategise, 4) Design, 5) Exit Strategy, 6) Select, 7) Negotiate	7

(Continued)

(Continued).

Authors and title	Constructs in the Study	Total
Jamshidi, P., Ahmad, A., Pahl, C., "Cloud migration research: a systematic review"	1) Feasibility Study, 2) Requirement Analysis, 3) Decision on Provider, 4) Sub-system to be Migrated, 5) Migration Strategies Development, 6) Code Modification, 7) Architecture Recovery, 8) Data Extraction, 9) Architecture Adaptation, 10) Test, 11) Deployment, 12) Training, 13) Effort Estimation, 14) Multi-Tenancy, 15) Elasticity Analysis, 16) Decision on Cloud Services, 17) Transformation, 18) Migration Validation, 19) Security Analysis, 20) Organisational Change	20
La & Kim, 2009, "A systematic process for developing high quality saas cloud services. Cloud computing"	1) Requirement Gathering, 2) Domain Analysis, 3) Architectural Design, 4) Database Design, 5) Implementation, 6) Testing, 7) Deployment, 8) On-Demand Deployment, 9) Dynamic Load Balancing	9
Meiländer, Bucchiarone et al., "Using a lifecycle model for developing and executing real-time online applications on clouds"	1) Early Requirement Engineering and Design, 2) Construction, 3) Deployment & Provisioning, 4) Enact Adaptation, 5) Identify Adaptation Strategy, 6) Operation & Management, 7) Deployment-Time Adaptation, 8) Run-time Monitoring	8
Varia, J., "Migrating your existing application to the AWS cloud. A phase-driven approach to cloud migration"	1) Cloud Assessment, 2) Understand Various Storage Options, 3) Application Migration, 4) Leverage the Cloud, 5) Optimisation	5
Wu, J., et al., "Migrating a digital library to a private cloud"	1) System Requirements, 2) Cost Analysis, 3) Resource Allocation, 4) System Compatibility, 5) Migration Plan, 6) Preparation, 7) Test, 8) Redundancy and Data Backup, 9) Configuration, 10) Security, 11) Backward Availability, 12) Post Migration	12
Betts, D., Homer, A., Jezierski, A., Narumoto, M., Zhang, H., "Moving applications to the cloud on microsoft windows azure"	1) Options for Hosting the Data, 2) Elasticity and Scaling, 3) Session Support, 4) Integrating with On-Premises Services, 5) Test and Production Environments, 6) Deploying the Web Application, 7) Transient Fault Handling, 8) Management and Monitoring	8
Maenhaut, p., Moens, H., Ongenae, V., Turck, F "Migrating legacy software to the cloud: Approach and verification by means of two medical software use cases"	1) Selecting Components, 2) Determining Provider Compatibility, 3) Determining Impact on Client Network, 4) Scaling the Application, 5) Decoupling Databases, 6) Adding Tenant Configuration Database, 7) Providing Tenant Configuration Interface, 8) Dynamic Feature Selection, 9) Managing Tenant Data, Users, and Roles, 10) Mitigating Security Risks, 11) Migration Cost	11
Kwok, T., "A software as a service with multi-tenancy support for an electronic contract management application"	1) Execution Analytics, 2) Coordinated Adaptation, 3) Feedback-Driven Development	3

Appendix B

An excerpt of the questions was used during the analysis of the retrospective case studies.

Interview questions

Questions related to cloud migration objectives

- Can you describe the cloud migration project case and as-is situation of the application is to be migrated to cloud?
- Can you describe what drivers (technical and none-technical) motive your organisation to make your application cloud-enabled?

Questions related to enacted migration method in the project

- What tasks did you perform during the *Plan Phase* of the migration project? Include specific examples of each.
- What work-products did you produced during the *Plan Phase* of the migration project? Include specific examples of each.
- What tasks did you perform during the *Design Phase* of the migration project? Include specific examples of each.
- What work-products did you produced during the *Design Phase* of the migration project? Include specific examples of each.

Overall questions related to challenges during the cloud migration process

- From what aspects do you think *Plan Phase* of a migration method differ from existing traditional methods? In other words, in the cloud migration process what aspect need to more emphasis?
- From what aspects do you think *Design Phase* of migration methodologies differ from existing traditional methods? In other words, in the cloud migration process what aspect need to more emphasis?
- From what aspects do you think *Enable Phase* of a migration methods (process) differ from existing traditional methods? In other words, in the cloud migration process what aspects need to more emphasis?

Appendix C

The list of questions that the domain experts were asked about the metamodel for the evaluation.

Questions about the reference model (metamodel)

Dear Reviewer,

This document consists of two sections. The first section introduces our generic reference process model for moving legacy applications to cloud environments. The second section asks you a number of questions about the reference model.

Thank you once again for taking part in this research and accepting the review invitation for this study.

Demographic information about the reviewer

1. What is your affiliation?
2. What is your current position title?
3. How many years of experience do you have in legacy application migration to cloud environments?
4. Which of the following titles describes your IT role best?
 - a. Project manager
 - b. System analyst
 - c. Software developer (or programmer)
 - d. Software architect
 - e. Academic researcher
 - f. Process/Method Engineer
 - g. Other: please specify
5. What was the industry sector of the last legacy applications that were migrated to the cloud and you were involved in?
 - a. Energy
 - b. Financials
 - c. Health Care
 - d. Industrials
 - e. Information Technology
 - f. Materials
 - g. Telecommunication Services
 - h. Utilities
 - i. Consumer Discretionary
 - j. Consumer Staples
 - k. Other: please specify
6. migration to cloud environments?
 - a. Advanced
 - b. Good
 - c. Fair
 - d. Beginner
7. How many cloud migration projects have you been involved in so far?

Question related to the evaluation of the reference model/

1. Importance/relevance

Definition: by *importance/relevance* we mean that the proposed reference model should capture major generic and relevant constructs (phases, tasks, work-products, and principles) that are needed to be incorporated into all legacy migration processes. With respect to this definition, please answer the following five related questions a, b, c, d, and e.

- a. Does the reference model capture major generic constructs related to all cloud migration scenarios (Yes/No)? If not, please explain reason behind your answer. Please describe any missing constructs that you think should be included in the reference model.
- b. Please comment on how to improve the proposed model in reflection to missing constructs, if any.
- c. Please comment on any constructs that you think should not be included in the model and explain why?

- d. Does the reference model capture major relevant constructs related to all cloud migration scenarios (Yes/No)? If not, please explain reason behind your answer.
- e. Rate your opinion about the following statement: "Overall, the proposed reference model is complete in terms of inclusion of important/relevant constructs".
 1. Strongly agree
 2. Agree
 3. Neither agree nor disagree
 4. Disagree
 5. Strongly disagree

2. Correctness

Definition: By the term *correctness*, we mean that the reference model should define valid relationships between constructs as well as construct names and appropriate notational syntax. Considering this definition, please answer questions a, b, c, and d below:

- a. Are the model relationships (such as Produces, Uses, Follows, isAKindOf, and isAGroupOf) between the constructs appropriately decided on (Yes/No)? If not please provide reasons for your answer.
- b. Are the notations and syntax used to represent the reference model appropriate (Yes/No)? If not please explain reasons behind your answer.
- c. Are the connections between constructs and migration types appropriate (Yes/No)? If not, please explain reasons behind your answer.
- d. Rate your opinion about the following statement: "Overall, the defined relationships between constructs in the proposed reference model are correct".
 1. Strongly agree
 2. Agree
 3. Neither agree nor disagree
 4. Disagree
 5. Strongly disagree

3. Understandability

Definition. By the term *understandability*, we mean the clarity and meaningfulness of constructs in the proposed model for its audience such as process/method engineers, project managers, or software developers who are closely involved in a legacy to cloud migration scenario. Given that, please response to questions a, b, and c.

- a. Do the model constructs clearly convey their underlying meanings (Yes/No)? If no, please provide reasons for you answer.
- b. Are the model constructs names meaningful (Yes/No)? If not, please explain the reasons behind your answer.
- c. Rate your opinion about the following statement: "Overall, the proposed reference process model is understandable by its audience".
 1. Strongly agree
 2. Agree
 3. Neither agree nor disagree
 4. Disagree
 5. Strongly disagree

4. General Questions

- a. Describe the usefulness of the reference model document for its audiences. In what ways do you think the reference process model would create value to the audience? Please explain why.
- b. In your opinion, what are the *advantages* of the proposed reference model compared to models that you may have used in the past? Please provide a list of advantages and disadvantages of each.
- c. In your opinion, what are the *disadvantages* of the proposed reference model compared to models that you may have used in the past. Please provide a list of advantages and disadvantages of each.
- d. Would you use or offer the proposed reference model to guide IT teams to conduct legacy to cloud migration projects? Please provide reasons behind your response.